Artificial Ontogenesis:
A Connectionist Model of Development

Michael William Spratling

Ph.D.
University of Edinburgh
1999

Abstract

This thesis suggests that ontogenetic adaptive processes are important for generating intelligent behaviour. It is thus proposed that such processes, as they occur in nature, need to be modelled and that such a model could be used for generating artificial intelligence, and specifically robotic intelligence. Hence, this thesis focuses on how mechanisms of intelligence are specified.

A major problem in robotics is the need to predefine the behaviour to be followed by the robot. This makes design intractable for all but the simplest tasks and results in controllers that are specific to that particular task and are brittle when faced with unforeseen circumstances. These problems can be resolved by providing the robot with the ability to adapt the rules it follows and to autonomously create new rules for controlling behaviour. This solution thus depends on the predefinition of how rules to control behaviour are to be learnt rather than the predefinition of rules for behaviour themselves.

Learning new rules for behaviour occurs during the developmental process in biology. Changes in the structure of the cerebral cortex underly behavioural and cognitive development throughout infancy and beyond. The uniformity of the neocortex suggests that there is significant computational uniformity across the cortex resulting from uniform mechanisms of development, and holds out the possibility of a general model of development. Development is an interactive process between genetic predefinition and environmental influences. This interactive process is constructive: qualitatively new behaviours are learnt by using simple abilities as a basis for learning more complex ones. The progressive increase in competence, provided by development, may be essential to make tractable the process of acquiring higher-level abilities.

While simple behaviours can be triggered by direct sensory cues, more complex behaviours require the use of more abstract representations. There is thus a need to find representations at the correct level of abstraction appropriate to controlling each ability. In addition, finding the correct level of abstraction makes tractable the task of associating sensory representations with motor actions. Hence, finding appropriate representations is important both for learning behaviours and for controlling behaviours. Representations can be found by recording regularities in the world or by discovering re-occurring patterns through repeated sensory-motor interactions. By recording regularities within the representations thus formed, more abstract representations can be found. Simple, non-abstract, representations thus provide the basis for learning more complex, abstract, representations.

A modular neural network architecture is presented as a basis for a model of development. The pattern of activity of the neurons in an individual network constitutes a representation of the input to that network. This representation is formed through a novel, unsupervised, learning algorithm which adjusts the synaptic weights to improve the representation of the input data. Representations are formed by neurons learning to respond to correlated sets of inputs. Neurons thus became feature detectors or pattern recognisers. Because the nodes respond to patterns of inputs they encode more abstract features of the input than are explicitly encoded in the input data itself. In this way simple representations provide the basis for learning more complex representations. The algorithm allows both more abstract representations to be formed by associating correlated, coincident, features together, and invariant representations to be formed by associating correlated, sequential, features together.

The algorithm robustly learns accurate and stable representations, in a format most appropriate to the structure of the input data received: it can represent both single and multiple input features in both the discrete and continuous domains, using either topologically or non-topologically organised nodes. The output of one neural network is used to provide inputs for other networks. The robustness of the algorithm enables each neural network to be implemented using an identical algorithm. This allows a modular 'assembly' of neural networks to be used for learning more complex abilities: the output activations of a network can be used as the input to other networks which can then find representations of more abstract information within the same input data; and, by defining the output activations of neurons in certain networks to have behavioural consequences it is possible to learn sensory-motor associations, to enable sensory representations to be used to control behaviour.

Acknowledgements

I am greatly indebted to my supervisor, Dr. Gillian Hayes, for taking the risk of allowing me to commence a project that was initially rather vague and ill-defined and for her continued trust, and her support, encouragement and advice, while I developed my ideas.

Financial support was provided by the EPSRC.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

This thesis suggests that ontogenetic adaptive processes are important for generating intelligent behaviour. It is thus proposed that such developmental processes, as they occur in nature, need to be modelled and that such a model could be used for generating artificial intelligence (AI), and specifically robotic intelligence. Hence, this thesis focuses on how mechanisms of intelligence are specified. This specification level is just one level of analysis that can be used in the study of intelligence (table 1.1). Kenny (1989) identifies three others: intelligence itself (ability), the expression of the intelligence (exercise), and the mechanisms which implement the intelligence (vehicle)[1]. These levels of abstraction are useful for organising the analysis of intelligent systems, but should not be identified as being the same as intelligence itself (*e.g.*, reduction of intelligence to its vehicle is materialism, while reduction of intelligence to its exercise is behaviourism (Kenny, 1989)).

Aspect	Description	Discipline	Object of Study
Ability	The capacity to be intelligent	Philosophy	Mind
Exercise	The activity of being intelligent	Psychology	Behaviour
Vehicle	The machinery implementing intelligence	Physiology	Brain/Body
Specification	The development of mechanisms for intelligence	—	Genesis

Table 1.1: Aspects of Intelligence. Different levels of analysis which can be used in the study of intelligence. The specification level is of principal concern here, however, both the exercise and vehicle levels will be considered also. At the specification level both phylogenetic (evolutionary) and ontogenetic (developmental) processes are the object of study.

The specification level considers how the mechanisms implementing intelligence are developed. It is these mechanisms, in turn, which give rise to intelligent behaviour. Specification can thus be considered as a meta-level to the vehicle, which is itself at a meta-level to the exercise. It is necessary to consider both the exercise and vehicle levels in order to understand the requirements for the specification

[1] The ability, exercise, and vehicle levels are roughly equivalent to the computation, algorithm and implementation levels proposed by Marr (1985).

level. Ideally, the ability level should also be considered, however, intelligence itself is extremely hard to define. Since it is remote from practical considerations about the specification level, the ability level will be ignored and activity (at the exercise level) that appears to be intelligent will be considered sufficient. As well as studying these aspects of natural systems, further insights are provided by considering the requirements for the specification of artificially intelligent agents. Hence, this project has taken an integrative approach in which neurological, psychological as well as computational considerations have all been used to inspire and constrain a model of development. The resulting model is informed by biology and is roughly consistent with it, and is inspired by robotics and is roughly useful for it.

Artificial intelligence, as an engineering discipline, has the objective of creating intelligent machines. The specification of intelligence must thus be the principal concern for AI. Although this is true of all disciplines within AI that are concerned with creating intelligent artifacts only the sub-field of robotics is considered here. Currently, all methodologies for building robots predefine the rules of behaviour that the robot is to follow (section 2.3.1). There are problems with this approach (section 2.3.2).

1. **Tractability Problem.**
   Prespecification is too difficult for a complex robot/task/environment.
2. **Specificity Problem.**
   Prespecification is too specific to a particular robot/task/environment.
3. **Robustness Problem.**
   Prespecification is too unadaptive to changes in the robot/task/environment.
4. **Adaption Problem.**
   Adaption is too limited to create new abilities.

Not only is predefinition extremely difficult for a complex robot or one performing a complex task or operating in a complex environment, but the resulting system tends to be applicable only to the particular problem it was developed to solve and is brittle in the event of any unforeseen changes in its domain. Any learning that is provided requires the engineering of the task to be learnt, usually by providing appropriate representations of the input and output data to be associated, in order to allow a solution to be found (section 2.3.3). Learning, as currently used, thus provides a means of filling in the details of a predefined behaviour but does not generate any qualitatively new abilities, and hence is too limited to overcome the problems of prespecification. In contrast, the developmental process in animals does allow qualitatively new behaviours to be learnt, and biological systems demonstrate that a robust behavioural repertoire can be created tractably using non-specific mechanisms (section 3.3.2). Confusingly the word "development" has distinct meanings in artificial intelligence and in natural intelligence.

Artificial Intelligence uses an engineering science definition: design and implementation. The process through which a person produces an artifact.

Natural Intelligence uses a natural science definition: growth and qualitative change. The process through which a new born is transformed into a mature adult.

Phylogenesis provides specification of the animal's physiology, some innate abilities, and the mechanisms for the development of new abilities. These developmental mechanisms are strongly influenced by the constraints provided by evolution and by environmental factors. Development is thus an interactive process between genetic predefinition and environmental influences (section 3.3.1, Barlow, 1994; Elman et al., 1996; Plunkett et al., 1997). This interactive process is constructive: low-level competences (simple skills, and categorisations) are used to develop higher-level abilities (more sophisticated skills, and more abstract categorisations) (Arbib, Conklin and Hill, 1987; Elman et al., 1996; Quartz, 1999). The interaction of a situated agent with its environment guides the creation of cognitive machinery appropriate for representing and acting in that environment. Hence, not only is situatedness necessary for generating appropriate behavioural responses (Brady and Hu, 1994; Brooks, 1991b; Cliff, 1991; Hallam and Malcolm, 1994), situatedness is also necessary to develop those behavioural abilities in the first place (Clark and Chalmers, 1995; Dorffner and Prem, 1993).

Development has been mostly ignored in artificial intelligence (Rutkowska, 1996). This is a remarkable omission considering that it is the developmental process in biology that transforms the poor competence of the newborn into what is defined as intelligence (figure 1.1)[2]. If development was simply the maturing of innately predefined abilities (nativism), then there might be some justification for ignoring it, however, development is constructive and neglecting it in favour of design does not necessarily provide a 'short-cut' to producing intelligent machines: it may be easier to implement the mechanisms underlying development than to directly implement the mechanisms underlying intelligent behaviour.

Both reactive and deliberative behaviours are expressed by intelligent creatures: "behaviour is under the joint control of: (a) external stimuli; and (b) internal cognitions and goals" (Toates, 1998, section 3.1.1). Simple innate behaviours give rise to more complex behaviour through development. In the opposite direction, well practised deliberative behaviour becomes reactive through the process of automation (Posner et al., 1997; Toates, 1998; Arbib and Liaw, 1995; Karni, 1996). Simple, reactive, behaviours are triggered by direct sensory cues and hence involve non-abstract representations of the environment. However, direct sensory cues are not always available and more abstract representations are required to perform more complex behaviours (section 2.2.2). There is thus a need to find representations at the correct level of abstraction appropriate to controlling low-level and high-level abilities.

---

[2] There is a whole spectrum of levels of intelligence and the exact placement of different agents along this spectrum will depend on the measure for intelligence being used; there being no single objective test (Gregory, 1987). However, the level aimed for by AI is that of an adult human and it is for this reason that the Turing Test (Turing, 1950; Harnad, 1994) is the proposed measure of success. Some researchers have rejected the Turing Test in favour of a classification based on behavioural competence (Brooks, 1991a). However, all but the simplest creatures need to go through development before gaining behavioural mastery, and hence it is still adult level intelligence that is the goal.

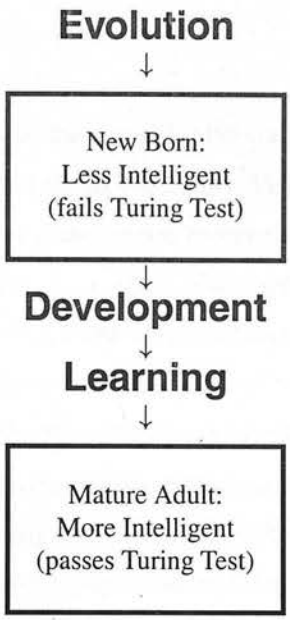


Figure 1.1: Stages in the specification of intelligence in nature. Ontogenesis is required to produce the level of intelligence aimed for by AI. However, AI attempts to generate adult level intelligence by direct prespecification (*i.e.*, without development).

In addition, it is not practical to learn to associate every possible sensory array with the appropriate set of actuator outputs; associations need to be formed at the appropriate level (Kirsh, 1991b; Felman, 1990; Davidsson, 1993, 1995; Intrator et al., 1995; Tsotsos, 1995; Solomon et al., 1999). The problem is not simply to calculate the mapping between inputs and outputs but to learn the appropriate inputs and outputs to map between. Hence, the task of development ought to provide a means of learning appropriate representations: finding appropriate abstractions and representations of an unlabelled world is "the essence of intelligence" (Brooks, 1991c).

Representations are found by recording regularities in the world or by discovering re-occurring patterns through repeated sensory-motor interactions (section 4.1.1, Bertenthal, 1996; Földiák, 1990, 1992; Wallis and Baddeley, 1997). By recording regularities within the representations thus formed more abstract representations can be found. Hence, simple, non-abstract, representations provide a basis for learning more abstract representations and appropriate levels of abstraction can be found by iteratively recoding information to represent more and more abstract features.

Distinct regions of the cortical sheet can be identified on the basis of anatomical and functional differences (Mountcastle, 1998, section 3.2.1). However, despite these differences, and the great diversity of functions attributed to the cortex, its overall structure is remarkably uniform (Crick and Asanuma, 1986; Ebdon, 1992), with region-specific variations being superimposed upon a standard micro-circuitry (Mountcastle, 1998) by uniform developmental mechanisms. The structural uniformity of the neo-

cortex has led many theorists to suggest that the cortex is also computationally uniform (Marr, 1970; Mumford, 1992, 1994; Ebdon, 1992, 1996; Phillips and Singer, 1997; Grossberg, 1999; Barlow, 1994; Douglas et al., 1989). The subdivision of the cortex into interconnected regions of functional specialisation suggests that it has a modular architecture, in which (along specific processing pathways, such as those through the visual areas) regions form a loose functional hierarchy (section 3.2.1, Felleman and Van Essen, 1991; Mountcastle, 1998) in the sense that lower-level representations are transformed into progressively higher-level, more integrated, representations (Ungerleider et al., 1998).

Structural changes in the cerebral cortex underly behavioural and cognitive development (section 3.2.3). Major structural changes occur at different times during the course of development in different regions (section 3.2.3, Jacobs, 1999; Shrager and Johnson, 1996). This is just what is required for a constructivist view of cortical development: regions which mature earlier provide a framework for those which develop later, allowing the later developing neurons to exploit the low-level representations formed in mature neurons in order to develop more abstract representations (Jacobs, 1999; Greenough et al., 1993; Shrager and Johnson, 1996; Singer, 1985; Elman et al., 1996).

Representations are stored by means of synaptic modifications in the neural circuits which will use this stored information (section 3.2.2, McClelland et al., 1995; Karni, 1996). Strong correlations have been found between the activity of individual neurons and behaviour (Newsome et al., 1989; Georgopoulos et al., 1986), and between the activity of individual neurons and sensory stimuli (Desimone, 1991; Perrett et al., 1992; Tanaka, 1996a,b). Hence, cell activity carries information about events ('event' will be used as a general term to refer to features of the environment *e.g.*, objects or actions). Cells near to the periphery represent low-level features of the sensory input. Cells higher up the hierarchy integrate information and respond to more elaborate 'features' of the input (Crick and Asanuma, 1986).

The pattern of activity on the neurons in a region constitutes a representation of the input to that neural network. The task for learning is to modify the synaptic weights to improve the representation of the input data. Inputs corresponding to features of the environment will be correlated and relatively independent from other sets of coactive inputs (section 4.1.1, Földiák, 1990; O'Reilly, 1998). Hence, re-occurring input patterns should be associated together (Földiák, 1992; Barlow, 1972, 1994; Edelman and Duvdevani-Bar, 1995)[3]. A neuron which forms such an association will thus respond to a more abstract feature of the data than those neurons which provide its input. This constitutes a local encoding of information that was previously represented by a distributed code of neural activities (section 4.1.2). Such a local code makes more explicit features within the input by transforming information encoded

[3] As well as finding spatial regularities, temporal correlations between inputs can also be used to find more abstract representations (section 4.2.2, Földiák, 1991, 1992; O'Reilly and McClelland, 1992; Wallis, 1994). Since the world generally changes slowly, there are likely to be regularities between consecutive sets of inputs (Wallis and Baddeley, 1997). By recording such regularities it is possible to classify input patterns which are dissimilar, but which have the same underlying cause, and hence to find representations that are invariant to viewpoint.

by the relationship between node activations into a representation at a single node. Associating correct responses to such an explicit representation is a trivial task, whereas associating correct responses to implicit representations is hard (Clark and Thornton, 1997; Thornton, 1994c, 1996a,b, 1995). Hence, recoding the data in this way makes tractable the task of learning sensory-motor mappings and the appropriate level of abstraction for each task is that at which this mapping can be learnt (not necessarily that at which the representation is purely local).

A connectionist model of development has been implemented. The model (chapter 4) consists of an assembly of artificial neural networks. Individual neural networks, called regions in analogy to cortical regions, consist of a single sheet of neurons in which all the nodes receive excitatory connections from an array of input sources. The input sources can come from environmental inputs or the outputs of other networks in the assembly. Networks are able to form distributed and local representations of their inputs, as appropriate to the structure of this data. In addition, since cells in cortical regions are often topologically organised the networks are able to form topological maps. This is achieved using a novel learning algorithm which can represent both single and multiple input features in both the discrete and continuous domains, using either topologically or non-topologically organised nodes. Both abstract and invariant representations can be learnt (chapter 5). Networks which receive inputs from the environment can find appropriate representations of this data. The output activations of such networks can be used as the input to other networks which can find representations of more abstract information within the same input data (section 6.1).

By defining the output activations of neurons in certain regions to have behavioural consequences it is possible to learn sensory-motor associations (section 6.2). Learning a new skill requires learning the relationship between the motor action and sensory cues. This mapping is learnt by finding appropriate connection weights from a sensor region to a motor region. The properties of the learning algorithm mean that such a mapping can be learnt without going through a distinct training phase. However, some simple, innate, abilities are required to under-pin the training (*e.g.*, for hand-eye coordination it is necessary that there is some fixation of the hand in order that the correspondence between arm position and gaze direction can be learnt). An assembly of two neural networks has been used to implement a simple skill by learning the mapping from a sensor network to a motor network. More complex mappings may be learnt by allowing nonlinear interactions between domains, *e.g.*, so that reactive behaviours can be gated by other sensory inputs. A model of a nonlinear neuron has been developed for this purpose (section 4.4.2). In addition, a mechanism to allow learning to be modulated by other inputs (section 4.2.4) has been used to allow coordinated behaviours to be learnt by using reinforcement signals derived from the environment.

Chapter 2

DEVELOPMENT AND ARTIFICIAL INTELLIGENCE

This chapter attempts to motivate the idea that a developmental perspective in robotics would be beneficial. It assesses the problems with current approaches to robotics, and considers the requirements for developmental mechanisms that can be used to constrain a developmental model. Many of these arguments may also be applicable to artificial intelligence in general, however no consideration is given to other areas of AI outside robotics.

Research in robotics is commonly classified as either knowledge-based or behaviour-based. These frameworks are often characterised as shown in figure 2.1. It is generally said that knowledge-based architectures are distinguished by being serial, disembodied, symbolic, programmed, deliberative and having a functional decomposition; while behaviour-based architectures are parallel, embodied, non-symbolic, emergent, reactive and have a behavioural decomposition. Most of these properties are not definitive to either approach: it is not essential for a knowledge-based architecture to be serial, nor are most behaviour-based robots purely parallel; knowledge-based robots are as much embodied as behaviour-based ones; behaviour-based architectures are symbolic, and are as much programmed as knowledge-based ones. Such a classification also fails to describe the many robot designs which are of neither type. Hence, rather than attempting to assess robotics by considering these two frameworks separately they will be analysed at the three levels of description introduced in table 1.1[1]. Each of these levels can also be used to determine constraints on a developmental model.

1. Exercise: the behavioural repertoire expressed by the robot.

The two extreme positions are reactive and deliberative behaviour.

[1] Excluding the ability level.

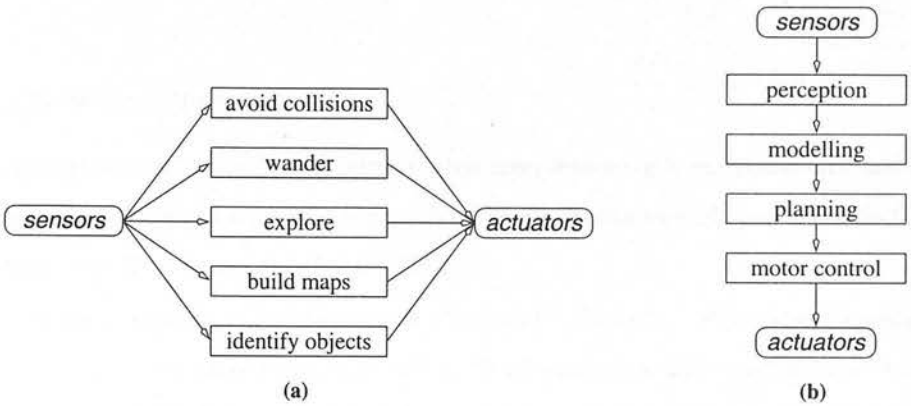


Figure 2.1: Standard caricatures of the behaviour-based and knowledge-based robot frameworks. (a) The behaviour-based robot framework. (b) The knowledge-based robot framework. (Adapted from Brooks, 1986)

This thesis argues that both reactive and deliberative behaviour ought to be provided.

2. Vehicle: the mechanisms which implement the behavioural repertoire.

The two extreme positions are parallel and serial architectures.

This thesis argues that a heterarchical architecture ought to be provided.

3. Specification: the creation of the mechanisms implementing the behavioural repertoire.

The only position is predefinition.

This thesis argues that development ought to be used. A system capable of development thus needs to be implemented.

2.1 Exercise of Intelligence

2.1.1 Types of Behaviour

2.1.1.1 Deliberative Behaviour

Deliberative behaviour is action in response to reasoning performed on an internal representation of the world. Knowledge-based robotics has demonstrated that such robots can reason about small domains but that complex environments, and low-level details, cannot be reasoned about in sufficient detail to control basic sensory-motor competences (Brooks, 1991c,b).

2.1.1.2 Reactive Behaviour

Reactive behaviour is action in direct response to sensory cues. Behaviour-based robotics (Brooks, 1991b; Maes, 1992) has demonstrated that considerable sensory-motor competence can be achieved

using predominantly reactive behaviour. However, high-level reasoning is not considered and there is little evidence to support the claim that this approach will scale-up from simple goal achieving skills to complicated behaviour (Pfeifer and Verschure, 1992).

Reactive behaviour relies upon "perfect sensor alignment" (Thornton, 1997), where a sensor input directly signals the event that should trigger an action. "Most reactive architectures actually amount to rule-based systems where the left conditions of the rules are required to come from physical sensors" (Dedieu and Mazer, 1991). However, in most cases sensors are not perfectly aligned but provide only partial information (the event has some impact on the sensor measurement) from which knowledge of the event can be inferred. In many cases integration of multiple sensor signals, possibly from different sensory modalities, might be required to provide sufficient data to control an action. Hence, more complex reactive behaviour is likely to require more abstract representations derived from the sensory data.

Reactive behaviour does not make predictions about the results of actions. This is in contrast to deliberative behaviour in which the expected outcome is compared with the robot's current goals.

2.1.1.3 Hybrid Behaviour

A hybrid system makes use of both reactive and deliberative behaviour. Either a deliberative reasoning system uses a reactive sub-system to execute plans and handle the low-level detail (*e.g.*, Malcolm and Smithers, 1991; Lyons and Hendriks, 1995; Hallam, 1991), or a reactive system uses a deliberative system to generate plans, and solve problems when the reactive system fails (*e.g.*, Mitchell, 1990; Chien et al., 1991; Krulwich, 1991).

Since neither purely reactive nor purely deliberative control seems to be adequate, combining the two would appear necessary (Sun et al., 1999). However, in most hybrid systems the reactive and deliberative parts remain as distinct, separate modules causing inflexibility and introducing interfacing problems. A single architecture capable of both reactive and deliberative control would be preferable.

2.1.2 Types of Knowledge

Reactive and deliberative behaviours demonstrate sensory-motor competences which are achieved by applying knowledge in different ways. Reactive behaviour, being triggered by sensory cues, makes use of implicit, non-conceptual, knowledge; while deliberative behaviour, being the result of reasoning, makes use of explicit, conceptual, knowledge. In either case the knowledge that is used could be at any level of abstraction. In general, simple behaviour can be achieved using non-abstract knowledge, while more complex behaviour makes use of more abstract knowledge. Furthermore, high-level, abstract, concepts are generally inappropriate to control low-level skills, while low-level, non-abstract knowledge is

inappropriate to perform complex behaviours. Abstract knowledge is required: at the perceptual level, to allow past experience to be applied to understanding novel events (Solomon et al., 1999; Kirsh, 1991b) (in order to see a shape as something rather than just a shape (Edelman, 1997) and to recognise similar states-of-affairs even if they have very different perceptual appearances (Davidsson, 1993, 1995)); and at the conceptual level, to make use of past experience to form expectations and make inferences about the current situation (enabling behaviour to not be simply determined by the current situation but to be influenced by past experience, and by plans and predictions about the future behaviour of the environment and other agents) (Kirsh, 1991b).

Hence, abstract and non-abstract representations are appropriate to different levels of behavioural complexity, but are inappropriate to other levels. It would thus seem to be necessary to use an architecture capable of representing both non-abstract and abstract knowledge so that the appropriate level of abstraction can be applied to each behaviour.

2.2 Vehicle of Intelligence

2.2.1 Types of Mechanisms

2.2.1.1 Serial Architecture

A serial architecture in which control is concentrated in a single process (*e.g.*, the Sense-Model-Plan-Act framework (Malcolm et al., 1989)) implies deliberative behaviour. The serial order of processing steps means that the sensor data is processed before motor actions are formulated and hence motor response is remote from the sensor input (and hence less reactive). The separation of the motor control system from the sensory system means that behaviour is formulated in response to the output of the sensor system which therefore constitutes an internal representation of the world. Sensing and acting can thus be considered as peripheral tasks whose role is to simply perform transformations to and from the internal representations. A single pathway from sensor input to motor output requires the internal model to be comprehensive in order to control all behaviours. In addition, since the actions which are appropriate to the current situation cannot be known until the sensor data has been processed, the sensor system must process all data which may potentially be required. In practice this usually means that the internal representation is a reconstruction of the external world (Edelman, 1994). However, the essence of a model ought to be that it is a simplification, otherwise nothing has been gained in creating it: there is a trade off between its faithfulness to reality and its practicality and usefulness (Heylighen, 1991). Also, such a sensing system will be passive, implementing the same general methods regardless of the context. Passive and active sensory systems are compared by (Rimey and Brown, 1992) and the passive approach is criticised by (Sloman, 1989; Brookes, 1992). The passive conversion of sense data into an

internal representation comes dangerously close to implying the need for a homunculus which views this internal scene rather than the world itself. In practice it is very difficult to extract a sufficiently accurate model in real-time, and in a dynamic world (and the actions of the agent ensure the world will not be static) the cost of maintaining correspondence between the internal representation and the world becomes prohibitive (Ballard, 1991). The generation of the world model, rather than competent behaviour, then becomes one of the principal tasks of the agent.

2.2.1.2 Parallel Architecture

A parallel architecture in which control is distributed between separate processes (*e.g.*, the behaviour-based approach (Malcolm et al., 1989)) implies reactive behaviour. The execution of processes in parallel means that control decisions are taken independently. Control is distributed among the modules. Hence, coherent behaviour can only arise through the carefully planned interactions of the various component processes among themselves and with the physical world (Brooks, 1991b). The control strategy is not explicitly encoded in a single module but is implicit in the design and interaction of the behaviour modules (Braitenberg, 1984; Brooks, 1991c). Such coordination may be provided through the mutual inhibition of incompatible behaviours (Brooks, 1994), message passing between modules (see figure 7 in Brooks, 1985) and action selection schemes (Maes, 1991; Werner, 1994) (hence few architectures are purely parallel, and some 'parallel' architectures may even include a single control module).

Since each parallel controller connects sensor input to motor output, perception is divided among many modules and "there is no identifiable place where the 'output' of perception can be found" (Brooks, 1991c). A comprehensive world model is not useful as it would have no impact on modules other than the one that generated it. At most, each module will generate a simple model of those aspects of the world which influence its behaviour and hence representations will be closely linked to sensing and movement. Passive reconstruction of the world can be avoided by using sensor data directly: the robot is situated in the real world and makes active use of its environment, referring to its own sensors rather than to an internal model of the world (the world is used as its own model) (Brady and Hu, 1994; Brooks, 1991b; Cliff, 1991; Hallam and Malcolm, 1994). This has inherent advantages in that the accuracy of the model is assured, and the dynamics of the world do not create the need for the constant updating of internal representations. However, direct responses to sensor data (*i.e.*, reactive behaviour) mean that the environment has been factored into a set of indicators for appropriate behaviours (Kirsh, 1991b) and this choice of sensory cues still forms an implicit (distributed) model of the world.

2.2.1.3 Hierarchical Architecture

Hybrid behaviour (section 2.1.1.3) may be generated using a combination of a fairly standard behaviour-based and knowledge-based systems. By splitting the problem between component sub-systems the

most appropriate level of control can be applied to different problems and the task of designing the system may be made easier (Hallam, 1991). The sub-systems may interact solely by exchanging information about the current state-of affairs (Malcolm and Smithers, 1991; Lyons and Hendriks, 1995). Alternatively, the behaviour of one system might be modified in response to the behaviour of the other: either (top-down) by using the knowledge-based system to formulate new stimulus-response rules to deal with situations in which the reactive system fails (Mitchell, 1990; Chien et al., 1991; Krulwich, 1991), or (bottom-up) by using the knowledge-based system to abstract rules from the successful performance of the reactive system (Sun et al., 1996, 1999). Other architectures use more levels of control (most commonly three levels which are often described as reflexive, planning and modelling layers) (Albus, 1981; Ferguson, 1992; Hexmoor et al., 1993; Klingspor and Morik, 1995; Roitblat, 1991; Hallam, 1991). As with the two level architectures the layers are modifiable in only some of these models and the training is similarly unidirectional: either top-down (*e.g.*, compiling planned actions into reactive rules (Hexmoor et al., 1993)), or bottom-up (*e.g.*, to generate reasoning rules by abstraction (Klingspor and Morik, 1995)). Rather than using a predefined number of levels in the hierarchy other architectures learn to decompose tasks into however many layers of control are required (Riekki and Röening, 1995; Digney, 1998).

A hierarchical architecture would seem appropriate to implement both reactive and deliberative behaviours, such that the correct type of reasoning can be applied to each situation (section 2.1.1.3). However, a sharp division between the reactive and deliberative components, as is used in most models, introduces practical problems and is biologically implausible (Yamauchi and Beer, 1994). A hierarchical architecture also allows the building of useful representations of the world, but does not imply the use of a comprehensive, internal, world model.

2.2.2 Types of Representation

Serial and parallel control architectures tend to utilise different types of representation. A serial architecture requires an explicit, monolithic, world model while a parallel architecture can use an implicit, distributed, world model. Both are forms of internal representation of the world: "all environmental properties which feature within [any] agent's behavioural contingencies have an internal representation within the agent" (Thornton, 1994a). Representation is unavoidable; it occurs wherever something may stand in for something else (Thornton, 1994a; Bechtel, 1996). Hence, behaviour-based robots use representations (Clark, 1994; Thornton, 1994a; Vera and Simon, 1993) despite many claims to the contrary (Brooks, 1991c, 1985, 1986). Brooks' arguments against representation appear to actually be against explicit, abstract, conceptual representation and exhaustive, detailed, monolithic world models (Kirsh, 1991b). Rejection of an exhaustive, detailed, monolithic world model seems to be justified (Edelman,

1994)[2]. However, the use of representation need not imply the use of a detailed world model, nor behaviour that is exclusively deliberative (Thornton, 1994a). Hence, although representation has been found necessary in general (König et al., 1998; Markman and Dietrich, 2000) there is disagreement about the properties of these representations (Markman and Dietrich, 2000). Such disagreement stems from the implicit assumption that all representations must have the same properties, and hence if one cognitive process can be shown not to require certain properties, then no cognitive process can require such representational properties (Markman and Dietrich, 2000). There seems to be little evidence to support such generalisation; on the contrary, it seems likely that different representational properties will be required for different cognitive tasks (Markman and Dietrich, 2000).

Reactive robots have demonstrated that abstract representations are not necessary for low-level skills, however, abstract representation does seem to be required for high-level behaviour. Without abstraction a reactive agent's repertoire of behaviour is limited to simple reflexive actions in response to a finite set of stimuli represented directly in the sensory input (Patel and Schnepf, 1992). More abstract representations are needed when information cannot be obtained directly from observation (*e.g.*, for tasks in which sensors are not perfectly aligned and hence direct sensory cues are not available (Thornton, 1997, see section 2.1.1.2)). In addition, more abstract representation provides a means of resource-saving for agents in structured environments (Thornton, 1994a): efficient systems "convert every frequently encountered important situation into one of 'virtual stimulus response' in which internal state (intention, memory) and sensory stimulus together form a compound stimulus that immediately implies the correct next intention or external action" (Wilson, 1991). Complex behaviour results from a response to an abstract, internal, representation while simple reactive behaviour results from a response to a non-abstract, external, stimulus.

Without abstract representations it would be necessary to associate every possible set of sensory inputs directly with the appropriate set of actuator outputs. It is not practical to do so since there would be too many such mappings (*e.g.*, to respond to all sensory patterns consisting of pixel pairs on the retina would require every combination of input pairs to be mapped to a motor response; a number, $(10^6)^2$, greater than the number of neurons in the cortex (Felman, 1990)). Not only is it impractical to store these mappings it would also be difficult to predefine or to learn (Intrator et al., 1995; Tsotsos, 1995)) such a large number of stimulus-response pairs. Instead, more abstract representations must be used to reduce the volume of information that needs to be recorded (and defined or learnt) and provide some generalisation between similar situations. However, an appropriate level of abstraction must be used. It would be equally impractical to represent every possible combination of events as a single concept

[2] There is no need for a monolithic world model when it is possible to refer directly to the world for the required information (Ballard et al., 1997). In certain situations when such information is not available it is possible to manipulate the world to record the required data (*e.g.*, by making notes or by using pen and paper to do long division), in this way internal state or memory can be transferred to the physical properties of the world (Clark, 1997, 1999a,b; Kushmerick, 1997).

(*i.e.*, a 'grandmother cell' representation in a neural network). The appropriate level of abstraction will vary between tasks and hence it would seem necessary that a range of levels be possible. This range of levels of representation is required to implement the range of knowledge types found necessary in section 2.1.2.

2.3 Specification of Intelligence

2.3.1 Methods of Specification

2.3.1.1 Knowledge Engineering

This approach has its roots in the symbolic processing approach to AI. As such it attempts to implement high-level cognitive abilities, using high-level, symbolic programming. The aim is to extract, formalise, and encode human knowledge. The underlying assumption is that agents can be formally specified and hence studied abstractly at the knowledge level (Kirsh, 1991a). In order to logically reason about the world it is necessary to simplify the environment.

2.3.1.2 Behaviour Engineering

This approach has its roots in the 'non-symbolic' approach to AI. As such it attempts to implement low-level cognitive abilities, using the interaction between many simple processes. It is generally only possible to design the interaction between a limited number of processes (Smithers, 1997) which limits the complexity of tasks that can be achieved.

It is often argued that the intelligence of such a system is an emergent property. It is the case that global behaviours can emerge from interactions between the component parts, without such behaviours being properties of individual components. (In this case 'emergent' is used in the sense of arising out of the system, as a natural consequence.) However, emergence is often used to imply that the global behaviour is less predefined than it would be if programmed explicitly. This is not the case. Such behaviour must be just as carefully designed, otherwise it could not be arranged to occur in any principled manner. (In this case 'emergent' is being used in the sense of arising unexpectedly, as an unpredicted consequence (Umerez et al., 1993), but if that were the case then behavioural engineers could not even aim to make robots with competent performance.) The "success of the behaviour-based approach to robot control can be attributed largely to the skill of the designer in selecting the right conditions and action parameters for each behaviour process and the right interprocess connections" (Pebody, 1995). Behaviour is predefined by the designer through the specification of:

- the behavioural decomposition necessary to achieve the required task;

- the sensory signals which decompose the environment into cues for behaviours;
- the reactive-rules which specify the motor responses to sensory cues;
- the coordination and integration of the behaviours to produce coherent and relevant action, and mechanisms to resolve conflicts between modules.

2.3.1.3 Artificial Evolution

Evolutionary mechanisms in nature have succeeded in specifying control systems for biological creatures. Artificial evolution has thus been applied to designing robot controllers. Such methods are used to evolve neural network controllers, either by finding appropriate (hard-wired) neural network topologies and connection weights (Cliff et al., 1992), or by finding appropriate learning parameters for predefined network topologies (Floreano and Mondada, 1996). However, the search space for complex controllers is very large, resulting in many potential solutions needing to be evaluated. In addition, the time required to evaluate, even in simulation, complex behaviours will be excessive, which means that this method remains limited to developing very simple agents.

The search space can be reduced by using constraints on the architecture that is to be evolved. For instance, the problem might be decomposed (by the developer) and evolution used to design the components parts (Perkins and Hayes, 1996) or the length of the required genotype reduced by using an indirect mapping from genotype to phenotype so that more complex neural networks could be encoded (Dellaert and Beer, 1994; Kodjabachian and Meyer, 1994) (for instance by generating repeated neural structures). However, although such methods may reduce the search space they do not reduce the time required to evaluate the fitness of potential solutions. Evolution also does not overcome the other problems of predefinition discussed below.

2.3.2 Problems With Predefinition

The three methods for designing robots all specify the rules of behaviour that the robot is to follow and only differ in how the rules are represented and defined (figure 2.2). Hence, at the specification level of analysis, there is little significant difference between current robot frameworks.

The specification of robot control through predefined rules of behaviour results in the following four inter-related problems.

1. **Tractability Problem.**

   Prespecification is too difficult for a complex robot/task/environment.

2. **Specificity Problem.**

   Prespecification is too specific to a particular robot/task/environment.

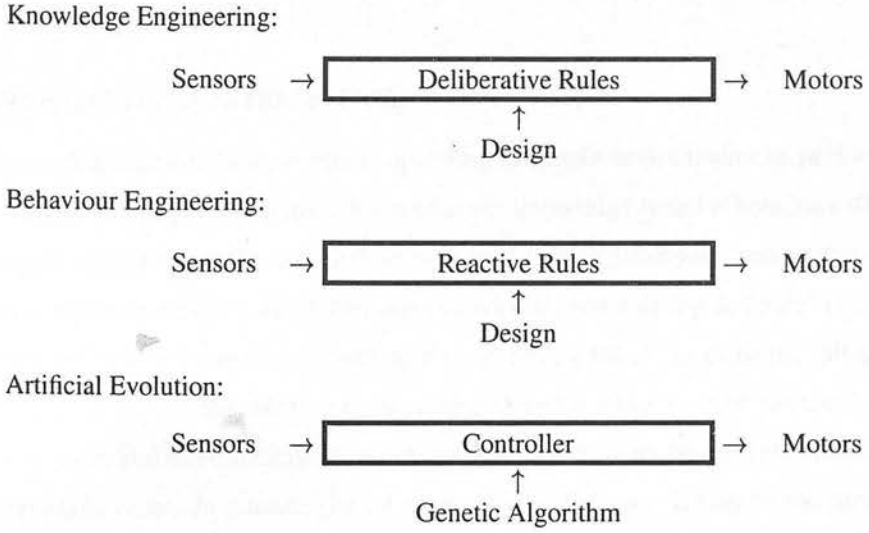


Figure 2.2: The similarity between different methods for building robots. All methods prespecify the rules of behaviour that the robot is to follow.

3. **Robustness Problem.**

   Prespecification is too unadaptive to changes in the robot/task/environment.

4. **Adaption Problem.**

   Adaption is too limited to create new abilities.

The first three of these problems are directly due to over-reliance on predefinition. All of these problems might be alleviated by the use of learning. However, commonly applied learning techniques are insufficient to completely solve these problems and this results in the fourth point, which is considered in section 2.3.3. Essentially, learning as it is currently used is insufficient to provide autonomy. Predefinition results in automatic (self-acting, involuntary) control systems. Automatic agents are self-regulating but they follow predefined rules. In contrast, autonomous (self-governing, independent)[3] agents can adapt the rules they follow and create new rules for behaviour[4]. Creating new rules for behaviour is exactly what development does. Hence, development is just the sort of process required for autonomy.

2.3.2.1 Tractability Problem

Prespecification relies on a significant design effort (whether this be through traditional symbolic programming or behaviour engineering) or on prolonged search and testing time (for evolutionary robot-

[3] Autonomous systems form a more general class of self-regulating system since they create their own laws of regulation as well as regulating behaviour with respect to both self-made and innate laws (Smithers, 1997). However, the term 'autonomy' is often misused in AI resulting in its application to describe systems which are purely automatic or simply just mobile, to mean self-sufficient, or as a synonym for intelligence itself (Smithers, 1997; Gadanho, 1999).

[4] An autonomous agent must follow rules which govern the way in which it changes its rules of behaviour. Hence, at a meta-level it is also following predefined rules.

ics). The result is that only simple robots operating in simple environments to perform simple tasks can generally be developed using available resources: knowledge-based robots have difficulty scaling up to complex environments (Brooks, 1991b; Beer et al., 1991); behaviour-based robots have difficulty scaling to complex tasks (Brooks, 1997); and evolutionary robotics has difficulty scaling up to complex behaviours (due to increase in search space and testing time). In principle this problem can be solved through further predefinition (*e.g.*, by adding more 'situation specific circuitry' (Brooks, 1997) to cope with more and more situations). However, although it might be possible in theory to predefine all necessary behaviours, in practice (for all but simple applications) it may be too time consuming or too difficult. The difficulty of predefinition is due to the requirement that the designer foresee and take into account all eventualities (Smithers, 1997). In addition, controllers quickly become too complex to design or analyse. Furthermore, even if a tractable solution could be found so that an agent was purely predefined it would be inflexible and suffer from the problems discussed below.

Adaption could be used to help to make tractable the task of building complex robots. Learning could be used to fill in fine details and hence ease some of the burden on design. However, only a system which autonomously developed control strategies would completely solve the problem.

2.3.2.2 Specificity Problem

Predefining rules for behaviour produces *ad hoc* solutions to particular problems. The solution is specific to a particular robot morphology, acting in a particular environment to achieve a particular task. It may be possible to modify the rules to re-apply a control system to different applications, but this will result in new, highly specific solutions. Such a robot is inflexible and unable to cope with contingencies (or exploit opportunities) not foreseen by the designer.

Adaption could be used to enable controllers to be more generally applicable. Learning could be used to adjust parameters appropriate to different problems. However, only a system which autonomously developed control strategies would completely solve the problem.

2.3.2.3 Robustness Problem

The successful operation of a predefined controller relies on the developer having foreseen all potential situations. Since the controller is specific to a particular problem any deviation of the robot or the environment from that expected can cause failure. Prespecified rules are brittle in the face of deviations of the world from the abstractions used by the programmer (Clocksin and Moore, 1989).

Adaption could be used to make robots less brittle. Learning could be used for recalibration of a controller within the situation for which it was designed. However, only a robot that could adapt to situations significantly different from that for which it was designed would completely solve the problem and be able to repair deficiencies autonomously.

Development would be useful to make the process of specification tractable. In addition, the same processes responsible for development would be useful to allow an agent to operate successfully by continuing to adapt to the environment and learning from experience.

2.3.3 Problems With Learning

Learning approaches provide the robot with a strategy for discovering how to behave. In theory they have the potential to learn to solve any problem and to produce more generally applicable, and more robust solutions. In practice, learning is used only to fill in the details for explicit rules specified by design, or to learn skills for situations and their minor variations from examples chosen by the designer (Sun et al., 1996, figure 2.3). Hence, robots still have their control laws built-in, even if they are subsequently tuned by learning (Smithers, 1997). Learning is thus limited to providing quantitative changes in behaviour rather than developing any qualitatively new behaviour.

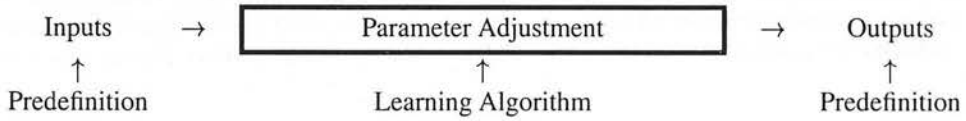


Figure 2.3: Learning is limited to parameter adaption within predefined behaviour capabilities. It provides only quantitative change in abilities.

2.3.3.1 Bias-Variance Problem

In any learning task there is a trade-off between bias and variance. A learning algorithm with high variance can potentially approximate many functions, however, the large search space may mean that it fails to find an appropriate solution to a particular problem (or fails to generalise from the training data). Biasing the algorithm will restrict the number of functions it can successfully model, hence, bias can be used to make it more likely that the required solution to a particular problem is found. Methods which automatically trade-off bias and variance have been developed, most of which adjust the structure of a neural network during training (Quartz and Sejnowski, 1997; Quinlan, 1998; Honavar and Uhr, 1993; Joseph, 1996; Fritzke, 1996). Progressive algorithms[5] intermittently increase the variance of a highly biased network (*e.g.*, by adding neurons) to improve accuracy. Regressive algorithms intermittently decrease variance of a highly unbiased network (*e.g.*, by removing unimportant neurons) to improve generalisation. There are two searches that need to be performed (Sazonov and Wnek, 1994):

[5] These are often called constructivist learning algorithms. However, the term constructivism is used in psychology to mean bootstrapping (using simple skills as a basis for learning more complex ones), while in these learning algorithms, as with all others, the goal is to approximate one particular function.

Relational Problem				Statistical Problem		
x_1	x_2		y	$\lvert x_1 - x_2 \rvert$		y
0	1	$\rightarrow$	1	1	$\rightarrow$	1
0	2	$\rightarrow$	0	2	$\rightarrow$	0
3	2	$\rightarrow$	1	1	$\rightarrow$	1
1	1	$\rightarrow$	0	0	$\rightarrow$	0
2	0	$\rightarrow$	0	2	$\rightarrow$	0
1	0	$\rightarrow$	1	1	$\rightarrow$	1
0	0	$\rightarrow$	0	0	$\rightarrow$	0

Table 2.1: Example of a relational and a statistical problem. In the relational problem there is no reason why any absolute value of any input variable should be correlated with the required output value. Hence, there is likely to be no strong statistical correlations between the input values and the required output value. For instance, $P(y = 0|x_1 = 0) = 0.5$ and $P(y = 1|x_2 = 0) = 0.33$. However, the sampling of a sub-set of the data may introduce spurious statistical dependences, *e.g.*, $P(y = 1|x_1 = 3) = 1.0$. The output value is actually dependent on the relative values of the inputs rather than on the absolute values. Since there are an infinite number of relationships that could exist between the inputs solving the relational problem is extremely difficult. Recoding the input data, in this example to provide an input which is the absolute difference between the original inputs, reduces the problem to a trivial task in which the input provides direct justification for the output. Such a statistical problem is easy to solve by observing the conditional probabilities. (Adapted from Clark and Thornton, 1997)

1. the search for the most suitable representation space;

2. the search for the most suitable representations within this space.

The first of these is the most fundamental (Quartz and Sejnowski, 1997).

In practice, in order to learn it is necessary to bias the learning procedure for the particular problem. This can either take the form of engineering the task to be well suited to the learning algorithm (by providing an appropriate representation of the input and output data) or biasing the learning algorithm to the particular task. Any learning algorithm with limited resources has a limited representational space and hence is biased to some extent; this implicitly predefines what can be represented (Quartz, 1993). However, the most common approach is to modify the training data for each particular problem and use general learning techniques. Such recoding reduces the problem to finding statistical regularities in the data: "a variety of learning algorithms tend to rely predominantly (and in some cases exclusively) on the extraction of a specific type of regularity from a body of input data. This type of regularity lies close to the surface of the training data, in the form of pronounced frequency effects and is thus fairly straight forwardly extracted by a variety of direct sampling methods" (Clark and Thornton, 1997). Such statistical learning problems are far easier to solve than relational problems (table 2.1; Clark and Thornton, 1997; Thornton, 1994c,b, 1995): statistical effects are explicit properties of the data while relational effects are implicit properties of the data (Thornton, 1996a). Implicitly encoded problems are unlikely to be solved unless the learning algorithm is biased, or the problem is recoded to provide an more explicit representation. "It is widely agreed that existing learning methods are extremely effective

provided the 'right' representation of the problem or environment is used. Thus constructive induction – the problem of finding the right representation – is a key challenge" (Thornton, 1996b). Hence, the real problem in learning is to find the correct inputs and outputs to associate. In other words it is necessary to find the appropriate representations for the inputs and outputs so that a standard, unbiased, learning algorithm can be used. It is a matter of finding the correct level of abstraction (section 2.2.2).

2.3.3.2 Ontological Problem

Finding appropriate representations is important both for learning (above) and for controlling behaviour (section 2.2.2; *e.g.*, the need for perfectly aligned virtual sensors which provide explicit representations in order to trigger behaviours). Hence, finding appropriate abstractions and representations of an unlabelled world "is the essence of intelligence and the hard part of the problem being solved. Under the current scheme the abstraction is done by the researchers" (Brooks, 1991c, 1986). Appropriate representations are thus imposed by design, in various forms:

- the symbols used in a traditional program;
- the behavioural decomposition in a behaviour-based program (Warrick and Fox, 1994);
- the provision of suitably processed inputs for learning algorithms (Manderick, 1991), and for supervised learning, the selection and classification of the training data.

"In most of our work, and in virtually all other research on robot exploration, knowledge of the properties of the robot's sensors and effectors, and of their relation with the properties of the environment, is provided by the researcher and embedded in the structure of the learning system" (Kuipers et al., 1993). However, such representations should not be imposed (Brooks, 1991c; Edelman, 1987), but rather learning appropriate representations ought to be a principal task of an autonomous agent. The major problem with current approaches to robotics is not the use of representation in itself (contra Brooks, 1991c,a), just the reliance on ascribed representations.

A representational system also faces the problem of being ungrounded. Proposed solutions to the symbol grounding problem (Harnad, 1990, 1993b; Ziemke, 1997) often rely on the ability of all representations to be ultimately:

1. derived from sensors (*e.g.*, the definition used by Connell and Mahadevan, 1993);
2. expressed in action (*e.g.*, the definition used by Brooks and Mataric, 1993).

Both these definitions attempt to make internal symbols represent tangible events in the external world. However, getting progressively closer to the raw data (*e.g.*, Warrick and Fox, 1994; Harnad, 1993a; Bonarini, 1996) does not seem to solve the problem since as soon as data is expressed as an internal

variable (however non-abstract) it is still a representation needing grounding. Grounding "is not limited to linking concepts to sensors and effectors, but also to any other internal mental state" (Dorffner and Prem, 1993). It is the problem of needing to decode internal representations that is the real issue, and since all robots use representations all potentially suffer from this problem, even those in which the correlation between the internal representations and external occurrences are learnt (contra Tani, 1996).

However, the use of a representation does not necessarily imply that this is something that needs decoding (Morasso et al., 1998). It is the relationship between representations that provides them with meaning (Duch, 1994; Edelman, 1996): "any time we learn a new concept we have to define it in terms of what we already know" (Duch, 1994). Representations are in general arbitrary symbols and it is the relationship between representations (second order isomorphism) that provides the similarity between a representation and the thing represented, not the similarity between the thing and the representation (first order isomorphism) (Edelman, 1998, 1996, 1994; Edelman and Duvdevani-Bar, 1995). Hence, as well as learning correlations between internal representations and external events, the representational system should also learn correlations between internal representations themselves.

2.3.3.3 Adaption Problem

In most applications learning mechanisms fail to provide much adaptability as they are not used to construct representations (Thornton, 1995), but only to perform parameter adjustment within predefined representations. The choice of task to be learnt, the choice of sensor and actuator system, or the predefinition of internal representations are often used to provide the learning algorithm with appropriate data. Hence, learning does not develop new behaviour but simply fills in the details of the mapping that has been implicitly defined in the design. In many applications, learning thus performs conditioning in which new triggers are found for a predefined action (Chang and Gaudiano, 1998, 1997; Pfeifer and Verschure, 1991; Verschure et al., 1992; Scheier and Pfeifer, 1995; Scheier et al., 1998). Reinforcement learning is also a common approach (*e.g.*, Sutton, 1991; Edelman, 1987; Almássy et al., 1998; Nehmzow, 1994; Colombetti et al., 1996; Fuentes et al., 1995; Ring, 1992; Sun et al., 1999; Humphrys, 1996; Connell and Mahadevan, 1993). Reinforcement learning provides the agent with 'reward' or 'punishment' during its performance. The learning mechanism attempts to apportion this reward to its previous actions with the aim of maximising its future reinforcement. An external teacher or an in-built value system (Rutkowska, 1997b) is thus needed to provide reinforcement. As with all learning mechanisms bias is necessary (in practice, if not in theory) to solve particular, non-trivial, problems. This can take two extreme forms: the value system may be simple but the behaviour has been innately defined so that reinforcement is simply used to learn to generate built-in behaviour at appropriate times (knowledge is innate to the behavioural system). Alternatively, the reinforcement function may need to be very elaborate in order to enable complex tasks to be learnt (knowledge is innate to the reinforcement

system). In the latter case the reinforcement policy effectively defines the exact behaviour to be learnt and the designer could have just as easily programmed the behaviour directly. One such example is when the reinforcement system needs to identify certain events in order to provide reinforcement to a system which is learning to identify those events. If such knowledge has already been predefined then there is no advantage in learning it. Supervised learning is another learning mechanism that suffers from this problem, requiring that knowledge be innately defined in the teaching system (Marcus, 1998).

Hence, learning is used to adapt existing behaviours to improve performance, but learning is not used to provide a way of developing new behaviours (Rutkowska, 1997a). A process of learning which did enable a robot to autonomously expand its behavioural repertoire would provide a mechanism for development. Learning algorithms are good at learning to associate inputs with outputs (finding mappings), and hence in most robotic applications learning is a case of associating cues derived from sensory data with appropriate motor responses ("generally speaking, the robot learning problem is to infer a mapping from sensors to actions given a training sequence of sensory input, action outputs, and feedback values" (Connell and Mahadevan, 1993)). But the problem is not simply to calculate the mapping between inputs and outputs but to learn the appropriate inputs and outputs to map between. Hence, the task of development ought to provide a means of learning appropriate representations.

Simple, non-abstract, representations can form the basis from which to learn more complex, abstract, representations. Simple, reactive, behaviours controlled by non-abstract representations can form the basis from which to learn more complex behaviours. Such a process can be applied recursively to keep increasing the behavioural complexity. Using simple skills to develop more complex ones (bootstrapping) is a good way, theoretically, to learn to solve difficult problems (Elman, 1993). Simple behaviours reduce the search space (provide bias) (Araujo and Grupen, 1996) for finding more complex ones.

2.4 Models of Development in Robotic Systems

2.4.1 Schemas

The term 'schema' is used to describe a unit of knowledge and the process by which that knowledge is applied (Arbib, 1995), whether it be perceptual, conceptual, behavioural or any other form of knowledge. Schemas are thus a very general concept used to describe many types of model, not only developmental ones.

A schema may be composed of a number of "smaller" schemas, and any one schema may also be involved in the implementation of several "larger" schemas. Thus a new schema can be created by the combination of current schemas to learn new capacities by building on current capacities, rather than learning from scratch (Arbib, 1995). A schema has an activation level describing its response

to the current pattern of sensory input or current activation of other schema (Arbib, 1995; Balkenius, 1992). The instantaneous pattern of activation within the schemas represents the current state of the environment and the agent, while the repertoire of schemas available and their structure represents the knowledge of the agent (Arbib, Conklin and Hill, 1987). Schemas cooperate and compete to perform computations (Arbib, 1995).

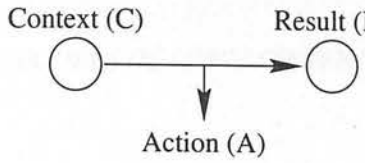
The concept has been widely used in psychology, neuroscience and artificial intelligence (Eysenck and Keane, 1990; Arbib, 1995). However, it is purely descriptive and while schemas provide a useful concept for reasoning about the computational mechanisms of intelligent behaviour they fail to specify algorithmic mechanisms by which these can be implemented. Hence various different implementations have been used, but these tend to often have a strong similarity to existing methods (*e.g.*, to behaviour engineering (Arbib, Iberall and Lyons, 1987; Lyons and Arbib, 1989; Arbib and Liaw, 1995; Shastri et al., 1999), or to neural networks (Balkenius, 1993)).

2.4.2 CLife

This approach (Riegler, 1994a,b) constructs cognitive mechanisms from fuzzy rule-based elements, referred to as schemas. Such a schema is triggered when the sum of matches with its fuzzy conditions exceeds a threshold. An activated schema can then call other schemas, set values, place its execution into the background, or stop the execution of the schema which called it. All actions can be interrupted by innate, reflexive, events. Schemas are created through both evolution and ontogenetic development. The ontogenetic process allows new schemas to be added, either at random or when no current schema's conditions adequately match the current context. The characteristics of new schemas are simply copies of existing schemas with randomly modified parameters (Riegler, 1996). Hence the developmental mechanisms are rather weak and the main burden of learning is on phylogenesis. This approach is primarily a technique for the evolution rather than the development of control systems.

2.4.3 Genetic AI

This work (Drescher, 1986, 1987, 1991) attempts to implement Piaget's theory of genetic epistemology (section 3.4.3) for development during the sensory-motor period (section 3.1.3). It uses schemas consisting of a context, action, and result (figure 2.4). "The schema asserts that if its context is satisfied, taking its action is expected to bring about its result" (Drescher, 1986). Each schema also has auxiliary structures; the reliability, the extended context, and the extended result. The reliability measures the likelihood of the result being produced by the action. The extended context and extended result both maintain statistics on every other signal in the system in order to detect any which might also need to be included in the context or the result. If such items are found a 'spin-off' schema is produced which



Reliability:	$P(R\|CA)$,	$P(R\|C\neg A)$.
Extended Context:	$P(R\|CAC_x)$,	$P(R\|CA\neg C_x)$,
	$P(C_x\|CAR)$,	$P(C_x\|CA\neg R)$.
Extended Result:	$P(R_x\|CA)$,	$P(R_x\|C\neg A)$,
	$P(R_x\|CAR)$,	$P(R_x\|C\neg(AR))$.

Figure 2.4: A Schema in Genetic AI.

is a copy of the current one but with an appropriately modified context or result. Each primitive action is provided with a blank schema, which is initially activated at random in order to generate training data for finding appropriate values for the context and result. Schemas can also assert synthetic data items (ones not bound directly to sensory data) to account for discrepancies between successful and unsuccessful results which can not be explained from the extant data. Such synthetic items enable abstraction away from the sensory data.

This method has only been applied to a very simple simulated agent existing in a world consisting of nine discrete locations. The need for each schema to maintain statistics on every other data item in the system means that there is a major scaling problem.

2.4.4 Sensory-Association-Motor Architecture

The Sensory-Association-Motor (SAM) architecture for agent learning (Aitken, 1994) is inspired by the gross anatomy of the cerebral neocortex; it has learning modules corresponding to sensory, motor and association areas, which are bi-directionally connected through excitatory signals (figure 2.5). This architecture can be expanded by linking further modules to any two existing modules. The architecture can thus be built up hierarchically. The architecture sits on top of a predefined control system, the inputs and outputs of which form the environment for the SAM architecture.

Each module implements the same learning mechanism as all other modules. Modules detect correlations in their input and come to represent such correlations so that similar inputs in future are more likely to induce the same output from the module. Both neural networks and classifier systems are suggested as suitable learning mechanisms. Common behavioural sequences will result in the sensor module learning the correlation between the sensory inputs and the internal representation of the motor signals. In addition, the motor module will learn the correlation between the motor signals and the internal representation of the sensory inputs. Hence, similar sensory inputs are likely to induce given motor outputs in future, and vice versa. The association module provides context to distinguish between behaviours that share more primitive sensory-motor sequences. As the motor and sensory modules become correlated, so the association module comes to record these correlations and its own outputs become included in the associations in the motor and sensory modules.

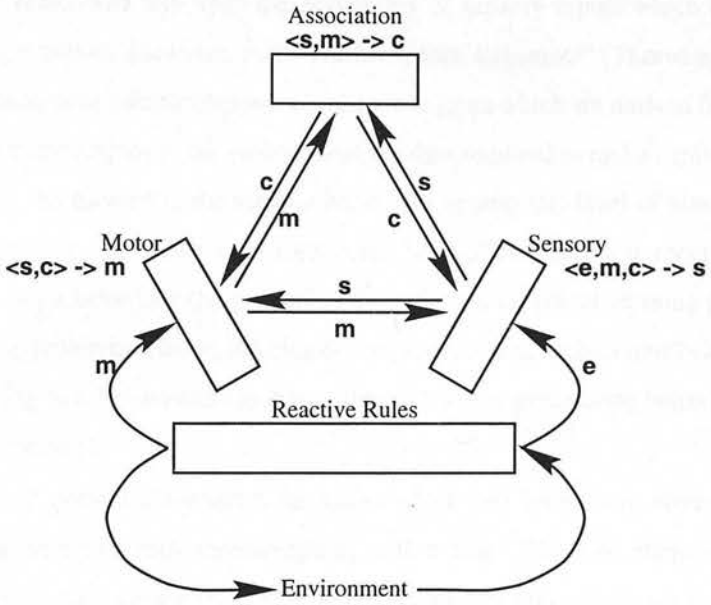


Figure 2.5: The Sensory-Association-Motor (SAM) architecture.

No results for this model have been presented. However, even if the SAM architecture was effective at learning representations of behavioural sequences it would fail to provide a useful model of behavioural development since it has no means of affecting the motor output. All the behaviour must thus be predefined (within the reactive rules module). There are also likely to be issues of timing which would need to be addressed in order for modules to be able associate corresponding values, and a means of dealing with conflicting inputs to the same module from different sources.

2.4.5 COG

COG is a project to build a humanoid robot (Brooks and Stein, 1993). The developers have discussed the usefulness of 'adding' development to COG (Brooks, 1997; Ferrell and Kemp, 1996; Brooks et al., 1998; Scassellati, 1998) but they appear to have not implemented any developmental mechanisms and are instead using *ad hoc*, hand coded, solutions to design each behaviour (Brooks and Stein, 1993; Marjanović et al., 1996; Scassellati, 1998). The competence of COG is incrementally increased by building more complex skills upon simpler ones, but this is purely a programme for robot design rather than a model of developmental progression. The question of how new abilities could be learnt autonomously and hence the issue of the mechanisms of development are not addressed.

2.5 Summary

Simple reactive behaviours rely upon the availability of sensory inputs which directly signal events that should trigger action. However, such "perfect sensor alignment" (Thornton, 1997) is not always available and hence more complex behaviours require triggers which are derived from the sensory input. More abstract representations of the environment are thus required to make explicit information which is only implicitly represented in the sensory input. An appropriate level of abstraction is required to control each behaviour. In addition, an appropriate level of abstraction is required to make tractable the task of learning a behaviour (by presenting the task as a statistical learning problem rather than a relational learning problem). Hence, learning new representations enables new behavioural capacities to be learnt. Learning new behaviours has many advantages over predefining behaviours and can provide an agent with autonomy.

Chapter 4 will present a connectionist model which can learn more abstract representations of sensory data and associate such representations with action. The next chapter reviews behavioural development in biological system to provide further inspiration and constraints for this model.

Chapter 3

DEVELOPMENT AND NATURAL INTELLIGENCE

The aim of this chapter is to review the role of development in animal intelligence, in order to derive inspiration and specify constraints for a model of development. As in the previous chapter this review will be structured using the three levels of description specified in table 1.1[1].

1. Exercise: the behavioural repertoire and changes to this repertoire.

Primarily findings from psychology.

2. Vehicle: the mechanisms which underlie behaviour and changes in these mechanisms.

Primarily findings from neurophysiology.

3. Specification: the mechanisms which underlie changes in the mechanisms underlying behaviour.

Developmental mechanisms.

3.1 Exercise of Intelligence

3.1.1 Types of Behaviour

Two broad classes of behaviour are commonly identified (Beer et al., 1991). For example: procedural and declarative; stimulus-response and cognitive; and automatic and controlled. Although definitions may vary slightly these classes essentially correspond to the reactive and deliberative categories used in robotics.

[1] Excluding the ability level.

3.1.1.1 Deliberative Behaviour

Deliberative behaviours have properties such as being: voluntary (spontaneous), conscious (contemplative), attention-requiring, slow (persistent), effortful, flexible, goal-directed, context independent, occurring in series (to avoid interfering with each other), and can be explained and communicated by the agent (Posner et al., 1997; Toates, 1998; Steels, 1993). Essentially deliberative behaviours are initiated by internal goals.

3.1.1.2 Reactive Behaviour

Reactive behaviours have properties such as being: involuntary, unconscious, attention-free, fast, effortless, inflexible, situation-determined, context dependent, occurring in parallel (without interfering with each other), and are not accessible for explanation by the agent (Posner et al., 1997; Toates, 1998; Steels, 1993). Essentially reactive behaviours are initiated by external triggers.

Examples of reactive behaviour include reflexes (simple, fast, with intensity and duration dependent on intensity and duration of stimulus); taxes (orientation responses); fixed action patterns (extended, temporal sequence of action, with intensity and duration not dependent on stimulus); and parameterised action patterns (modifiable fixed action patterns) (Toates, 1998; Balkenius, 1994).

3.1.1.3 Hybrid Behaviour

Although purely reactive behaviour can explain the abilities of human infants (Rutkowska, 1994) it does seem to be the case that adults (of many species) also make use of deliberative behaviours. Hence, "it must surely constitute an implicit assumption of almost all behavioural science that behaviour is under the joint control of: (a) external stimuli; and (b) internal cognitions and goals" (Toates, 1998). Individuals demonstrate both reactive and deliberative abilities and the type of control that is exercised in performing a particular task can vary depending on circumstances (Toates, 1998). As well as such short-term changes longer-term influences on the relative importance of each type of control, for a particular task, occur with development, experience and pathology. Development gives rise to more complex behaviours and more deliberative abilities. Conversely, behaviours requiring deliberation can become reactive after extended practise (a process of 'automation') (Posner et al., 1997; Toates, 1998; Arbib and Liaw, 1995; Karni, 1996), while still maintaining the potential for deliberative control in exceptional circumstances (Arbib and Liaw, 1995). Automation thus results in initially deliberative behaviours becoming reactions to (possibly complex) external triggers and internal stimuli. Extended practise also results in improved (more skillful) performance (Karni, 1996).

3.1.2 Types of Knowledge

The behaviour expressed reflects the knowledge that the agent possesses about the world and about itself. Reactive behaviours are described as being based upon procedural, dispositional, or implicit knowledge while deliberative behaviours reflect declarative, representational, or explicit knowledge (Toates, 1994; Prescott, 1994).

Alternatively, knowledge can be categorised into pragmatic knowledge (that used for controlling behaviour) and semantic knowledge (that used for perceptual categorisation) (Jeannerod, 1997; Balkenius, 1994). Conceptual knowledge derived from perceptual categorisation appears to be organised in a hierarchical structure (*e.g.*, the concepts; animal, dog, poodle). The category which is learnt first is called the basic-level concept (*e.g.*, dog). Superordinate (*e.g.*, animal) and subordinate (*e.g.*, poodle) concepts are learnt later (Neisser, 1987b; Longothetis and Sheinberg, 1996). The basic-level is generally that at which differences between categories are most distinct, perceptually or functionally, from one another (Neisser, 1987a). Categorisations thus have a perceptual basis reflecting the structure of the world. The stable structure of the physical environment enables stable representations to be extracted. Categories are formed by finding regularities in the world or by discovering re-occurring patterns through repeated sensory-motor interactions (Bertenthal, 1996). Scaffolding, the process in which adults manipulate the infant's interactions so as to foster new abilities (Rutkowska, 1995), and imitation, the process by which adults provide examples of abilities, increase the chance of the child being exposed to, and hence discovering, useful patterns in the environment.

Finding regularities in the world can be applied not only to sensory-motor events but to social and cultural experiences. Hence, the influence of others (*e.g.*, through observation and communication) and of culture in general (*e.g.*, through language and schooling[2]) might provide regularities to be learnt about in the same way. There is evidence to suggest that self-image is also generated during development, and that children build models to explain their own behaviour in the same way that they develop models to explain other people's behaviour (Gopnik, 1993). Since the agent itself is a part of its own environment (it can observe its own actions and the effects it has on the world), it seems natural that the agent should form concepts of itself (Griffin, 1990) in the same manner as it does other aspects of the environment. Development thus underlies the creation of useful representations of the world, of others, and of oneself.

The regularities of the physical world enable its segmentation, and the resulting categories enable further organisation of more complex categories in terms of those already discovered (Bertenthal, 1996; Rosenfield, 1992): "as infants perceive the same information repeatedly, the stored representations derived from these experiences become increasingly rich and abstract" (Eimas, 1994). The basic-level concepts (and hence the ease with which differences are perceived) can change with experience (So-

[2] Knowledge acquired through schooling may have been developed over a time span far longer than the life of the agent and so it is in a sense inherited, but through intellectual concepts rather than through genetics.

lomon et al., 1999) or be affected by linguistic categories (Longothetis and Sheinberg, 1996; Harnad, 1987b,a). Hence, 'low-level' knowledge (*i.e.*, from perceptual discriminations) enables higher-level knowledge (conceptual categorisations) to be abstracted, but once it has then the 'high-level' knowledge can influence the lower-level modifying it (Arbib, Conklin and Hill, 1987) and enhancing the perceptual abilities required to form distinctions between conceptual groups (Harnad, 1987b,a; Harnad et al., 1991). Similarly, low-level behavioural abilities provide a basis for learning more complex behaviour. Learning is thus a 'constructivist' process: higher-level competences are learnt from lower-level ones, and this process can be repeated hierarchically. Bickhard (1995) claims that constructivism is the only plausible theory of epistemology "and any research programme with epistemological aspirations, such as artificial intelligence, that ignores it will do so at severe cost".

3.1.3 Development of Behaviour

Although there is considerable disagreement about the details of when and how behaviours develop it is clear, in higher animals, that behaviour does develop; from simple abilities at birth to more complex abilities at maturity. Qualitative changes in behaviour occur in a fixed order of progression (Shultz, 1991). Piaget's observations of the behaviour of children during development provide a good description of the process. Piaget describes behavioural ability in terms of schemas: units of behaviour and knowledge which change through interaction with the environment and other schemas (section 2.4.1). Piaget identifies the following stages (Drescher, 1991; Piaget and Inhelder, 1966; Beard, 1969).

1 Sensory-motor. Initial schemas are simple, innate, reflexes. Transition from reflexive responses to the use of schemas for practical, action based, intelligence and immediate comprehension. Actions are not guided by thought. (Birth until 1 or 2 years.)

1.1 Primary circular reactions. Repetitive sequences of actions are performed. Actions are reflexive and stimulus triggered. Schemas admit to modification in response to experience (*e.g.*, after a few days the infant can more easily find the nipple since it will have learnt to turn its head in whichever direction the nipple touches its face). Visual schemas enable slowly moving objects to be tracked, stationary objects to be visually explored, and attention to oscillate between one object and another. There is no coupling between modalities (*e.g.*, between visual and tactile schemas, such that a dropped object will not be searched for visually). Conception of the world is purely in terms of sensory stimuli.

1.2 Coordination of primary schemas. Particular stimuli invoke particular actions. Schemas for different sensory modalities begin to inter-coordinate. The infant watches the movements of its own hand and learns to control them. Eventually the infant learns to reach for objects. The infant will turn to look at an object which touches the hand, and will move objects to bring them into

sight. Conception of the world becomes orientated around objects.

1.3 Secondary circular reactions. Exploration of the effects of actions. Repetition of actions to reproduce accidentally discovered effects (such as repeated banging of objects to make a noise). Such actions may be extended to other objects even if they fail to produce the same effects, but only a specific action is used and the infant is unable to generalise to other actions which will produce the same effects with different movements. Schemas are repeated (*e.g.*, to re-grasp an object) or their actions extended (*e.g.*, to continue tracking in the same direction even when motion becomes too fast to track). Action is still dependent on stimuli, an object that is covered with a cloth will not be reached for. Conception of the world includes some spatial awareness with respect to the body.

1.4 Coordination of secondary schemas. Known actions are applied to new situations. Allows familiar schemas to be used for new purposes (*e.g.*, other hand movements will be employed to search for and re-grasp an object, not just the repetition of the previously successful one). This enables a systematic exploration of novel objects through many familiar schemas (*e.g.*, shaking, biting, *etc.*), whereas previously a particular object would invoke a particular schema. Hence, actions become stimulus independent. Potential perceptions are integrated with current schemas (the infant will learn to remove and discard an obstruction to reach an object, or to shift head position to see around an occlusion, or to rotate an object to see the desired surface). Conception of the world includes pairwise association between objects.

1.5 Tertiary circular reactions. Repeated 'experiments' are performed to see what an object will do (*e.g.*, repeatedly dropping an object to observe the behaviour of the object as it falls). Deliberate variations in actions are made to achieve a desired goal. The infant learns the effects of object interaction: that one object can block the motion of another, that one object can be used to move another (*e.g.*, pulling a rug to bring the toy which is on it within reach), and how to orientate objects with respect to each other. Conception of the world includes awareness of autonomous behaviour and causal relationships between objects.

1.6 Mental visualisation. The child becomes capable of finding new means of action, not by physical groping and trial-and-error, but by mental simulation. An understanding of personal spatial location with respect to the world occurs and language starts.

2 Preoperative. Use of symbolic structures in set patterns. Motor actions and objects have become represented by symbols but there are no abstract concepts or internalised schema. (1 or 2 years until 8 or 9 years.)

3 Concrete-operative. Operative use of symbols but still dependent on objects (classifications, relationships, numbers). Thought can guide action but does not operate independently of action. (8

or 9 years until 11 or 12 years.)

4 Formal-operative. Abstract, hypothetical and deductive reasoning. Thought becomes independent from action. (11 or 12 years onwards.)

3.2 Vehicle of Intelligence

3.2.1 Types of Mechanisms

3.2.1.1 The Cerebral Neocortex

The cerebral neocortex is probably the most important brain structure underlying human intelligence. Understanding neocortical function is thus crucial to the understanding of the higher functions of the brain (Rolls and Treves, 1998). Knowledge of the anatomy and connectivity of the cortex is extensive. However, the detailed functioning of the cortex is less well understood.

The cortical sheet has a laminar structure which is conventionally classified into six distinct neural layers (figure 3.1). However, the number of distinguishable layers differs for certain regions which appear more functionally specialised (*i.e.*, layer IV can be sub-divided into three separate laminae in primary visual cortex[3], while there is no layer IV in the primary motor cortex (Miller, 1996; Crick and Asanuma, 1986)). Layers are described as: superficial (layer I), upper (layers II and III), middle (layer IV), or deep (layers V and VI).

Distinct areas or regions of the cortical sheet can also be identified from various anatomical criteria (figure 3.2 shows an example of one such classification), from functional differences[4], and from patterns of extrinsic connectivity (Roland and Zilles, 1998; Mountcastle, 1998). Despite these differences, and the great diversity of functions that the cortex performs, its intrinsic structure is remarkably uniform (Crick and Asanuma, 1986; Ebdon, 1992). Area-specific variations are superimposed on a standard micro-circuitry (Mountcastle, 1998), and are due, primarily, to the nature and distribution of the extrinsic connections to and from each area rather than intrinsic differences (Mountcastle, 1998).

The uniformity of the neocortex has led many theorists to suggest that the cortex is also computationally uniform and to introduce models of cortical processing (Marr, 1970; Mumford, 1992, 1994; Ebdon, 1992, 1996; Phillips and Singer, 1997; Grossberg, 1999; Barlow, 1994; James and Hoang, 1993; Miller, 1996; Fuentes et al., 1996; Rao and Ballard, 1999; Douglas et al., 1989). Cortical uniformity implies that data obtained from the study of many cortical regions could be used to constrain such models

[3] These sub-layers arise due to reorganisation during development (Callaway, 1998) and hence the specialisation of the primary visual cortex may be caused by the nature of the inputs it receives.

[4] Anatomically distinct regions generally correspond to distinct functional roles (Rolls and Treves, 1998; DeFilipe, 1997; Crick and Asanuma, 1986; Mountcastle, 1998)

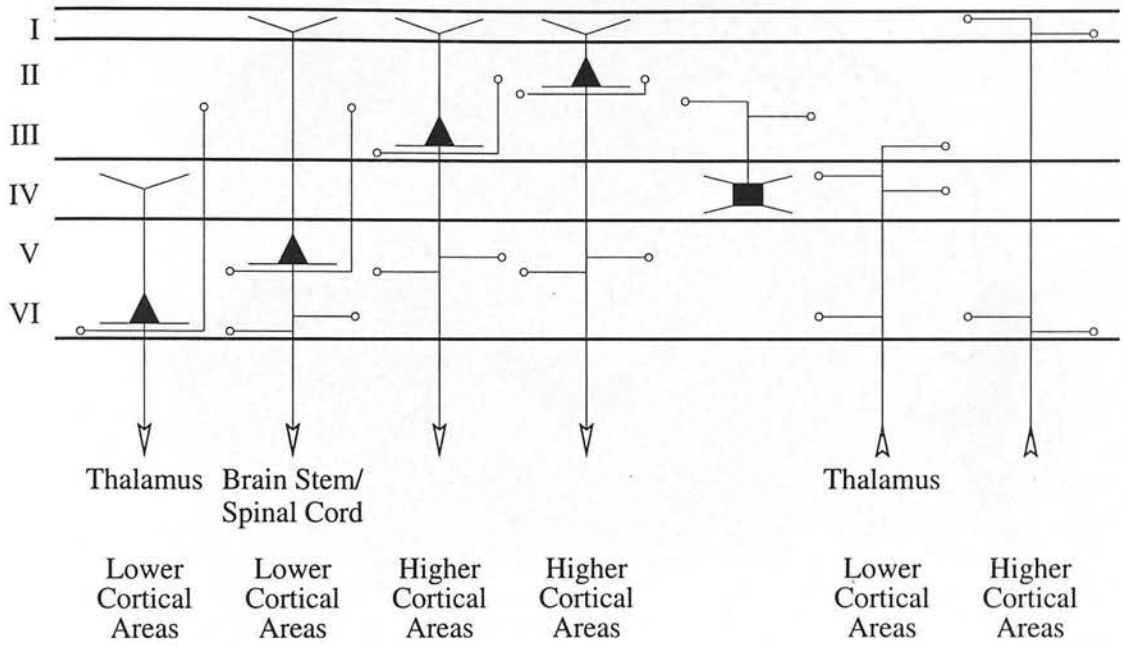


Figure 3.1: Cortical layers. A schematic of the layers of the neocortex, showing dendritic and axonal projections (axons have arrows and circles at terminations) for excitatory cells only (pyramidal cell bodies are triangular, spiny-stellate cell bodies are rectangular).(Adapted from Ebdon, 1996; Miller, 1996)

of cortical function (although primary sensory and motor regions are more specialised and hence may not provide general information about cortical circuits (Mountcastle, 1998)). In addition, evolution has maintained many aspects of neocortical circuitry (DeFilipe, 1997; Krubitzer, 1995) so that it is likely that data from many mammalian species could also be used to constrain such models. The size and location of corresponding regions vary considerably between species (Heiligenberg, 1991) (and can also vary between individuals of the same species (Mountcastle, 1998)). In addition, the inter-connectivity between regions, the extent of regions, the number of regions (and neurons), and the developmental timetable also vary between species and it is these differences, rather than novel cortical mechanisms, which account for differences in intelligence (Heiligenberg, 1991; Krubitzer, 1995; Mountcastle, 1998; Ebdon, 1992, 1996; Elman et al., 1996). However, the uniformity of the architecture of the cortex does not imply simplicity; the fine structure and the detailed connectivity of the cortex is complex. Experimental neuroscience has thus only been able to provide preliminary indications of how information is processed by the cortex (Rolls and Treves, 1998). Whatever the computational properties of cortical circuits they must be powerful in order to be applicable to such a diversity of functions across different species (Singer, 1995).

The human neocortex contains in the order of 10^{10} neurons. Neurons consist of a cell body (or soma), an input system (the dendrites) and an output system (the axon) (figure 3.3). Signals are generated in the cell body and transmitted along the axon. The axon may form in the order of 10^4 connections with other neurons. These connections between neurons are called synapses, and are the principal site at

Figure 3.2: Brodmann Areas. Cortical regions defined by cytoarchitectural (cell appearance) differences: lateral (top) and medial (bottom) views. (From Brodmann, 1914)

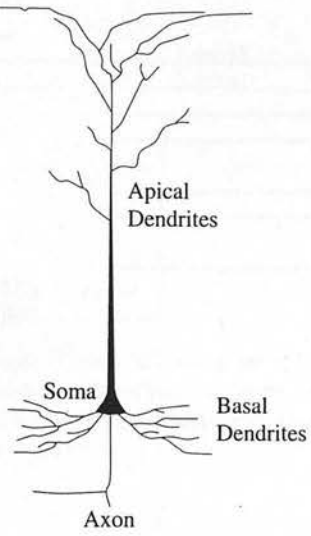


Figure 3.3: A Pyramidal Cell. Pyramidal cells are the predominant excitatory neuron of the neocortex. They consist of a long apical dendrite which extends in to the upper layers of the cortex, and basal dendrites closer to the cell body (soma), and an axon which projects downwards from the cell body. (Adapted from Calvin, 1996)

which information is transmitted between neurons and at which learning occurs (although other mechanisms of transmission and adaption exist). Two classes of neuron can be identified (1) spiny cells which form asymmetrical synapses, and (2) smooth or aspiny cells which form symmetrical synapses (DeFilipe, 1997; Mountcastle, 1998). The former are excitatory and the later inhibitory.

There are a large variety of inhibitory cell types (Mountcastle, 1998) which vary in morphology and preferred location within the cortical laminae. This diversity of forms suggests a diversity of functions (Somogyi and Martin, 1985; Houghton and Tipper, 1996). The location of the synapse on the target cell will also affect the function of the inhibitory signal: somatic inhibition and inhibition of the axon initial segment will suppress all excitation equally, and is hence non-specific to the source of excitation; dendritic inhibition will have most effect on the local section of the dendritic tree and hence will have a more specific effect on particular sources of excitation.

There are only two forms of excitatory cell. Spiny-stellate cells are confined to layer IV and lower layer III and are the main target for thalamocortical input fibres (Mountcastle, 1998). Pyramidal cells (figure 3.3) are the principal type of neuron making up 70-80% of the total number of neurons in the mammalian neocortex (Mountcastle, 1998)). In addition, pyramidal cells are a major source of information transmission within a cortical area (due to exuberant axon collaterals), and they are the only source of information transmission between different cortical areas, and out of the cortex to other parts of the nervous system (due to axons which descend into the white matter) (Mountcastle, 1998). In contrast, non-pyramidal cells only project within the local cortical area (Crick and Asanuma, 1986). The target for pyramidal axon projections without the local cortical area varies with the position of the cell within

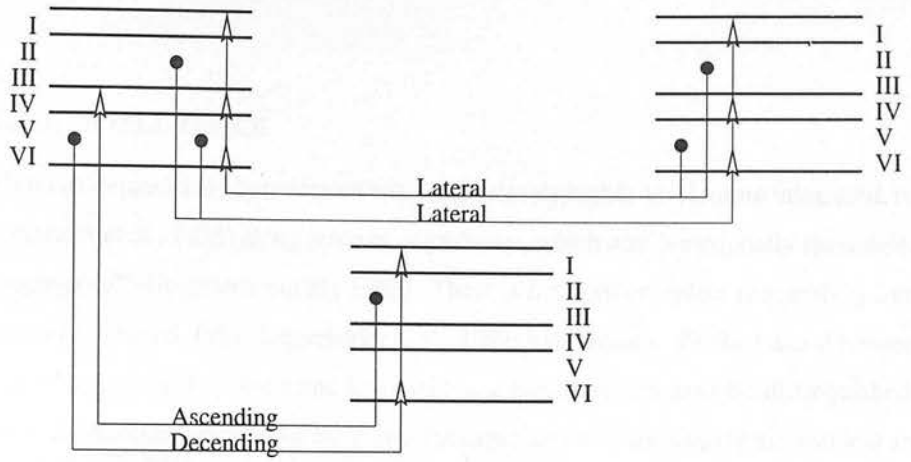


Figure 3.4: Cortical Pathways. Three cortical regions within a functional hierarchy are shown. One is at a lower-level in the hierarchy than the others and forms feedforward and feedback connections. The other two are at a similar level in the hierarchy and form lateral connections. (Adapted from Mumford, 1992; Felleman and Van Essen, 1991)

the cortical layers (figure 3.1). Projections from other cortical areas and subcortical structures also target particular layers depending on their source (figure 3.1). Pyramidals receive synapses on dendrites, soma, and axon initial segment (DeFilipe, 1997) with excitatory inputs mainly confined to dendritic shafts and spines and inhibitory inputs mainly to the axon initial segment and to the soma (Mountcastle, 1998). Pyramidal cells are unusual in having two, distinct dendritic trees: the basal dendrites spread out laterally and occupy the same layer as the cell body, while the apical dendrites ascend into more superficial layers. The apical dendrites ascend to layer I in most cases except for those of pyramidal cells in layer VI which only ascend to layer IV (Miller, 1996) and hence are unusual in receiving input from feedforward connections terminating in layer IV.

The subdivision of the cortex into interconnected regions of functional specialisation suggests that it has a modular architecture. Regions are connected via the axon projections of pyramidal cells which form long-range, excitatory, connections between regions (Ebdon, 1992; Crick and Asanuma, 1986)[5]. In general regions are reciprocally connected (Crick and Asanuma, 1986; Mountcastle, 1998). The layer of origin and of termination of these associative connections allows them to be classified as ascending (feedforward) or descending (feedback) (figure 3.4). The feedforward connections are provided by axon projections from pyramidal cells in layers II, III, and IV. These projections terminate predominantly in layer IV of the higher region (as do inputs from the thalamus). Feedback connections are provided by axon projections from pyramidal cells in layers V and VI and terminate in layers I and VI of the lower region (or are sent to subcortical structures). Hence, regions appear to form a loose functional hierarchy (Felleman and Van Essen, 1991; Mountcastle, 1998; van Essen and Deyoe, 1995) in the sense

[5] All inputs and outputs are excitatory, but such excitatory inputs may synapse onto inhibitory neurons in the target region to generate inhibitory effects.

that lower-level representations are transformed into progressively higher-level, more integrated, representations (Ungerleider et al., 1998) along streams or pathways which are "functionally specialised and anatomically segregated" (Ungerleider et al., 1998). There is far from complete connectivity between all cortical regions (Jeannerod, 1997; Ungerleider et al., 1998; Mountcastle, 1998). Lateral connections between regions at approximately the same level within a hierarchy can also be distinguished (figure 3.4). For central association regions of the cortex the organisation is less clearly hierarchical and the majority of connections are lateral (Mountcastle, 1998). While groups of regions can be in a hierarchical arrangement, it is not the case that the cortex forms a single hierarchy nor is there a single region which integrates all information (Mountcastle, 1998; Bartels and Zeki, 1998). Hence, the cortex might better be described as a distributed system rather than a hierarchical one. An example of distributed processing is the visual system (van Essen and Deyoe, 1995) in which two divergent streams can be identified: the ventral pathway to the temporal lobe which processes information about 'what' a visual stimulus is, and the dorsal pathway to the parietal lobe which processes information about 'where' a visual stimulus is (Ungerleider et al., 1998; Ungerleider, 1996; Sakata, 1996; Mountcastle, 1998; Bartels and Zeki, 1998; Fagg and Arbib, 1998; Wallis and Bülthoff, 1999), however, this may be an over-simplistic interpretation and there is still significant connectivity between these separate pathways (Jeannerod, 1997).

3.2.2 Types of Representation

Knowledge is stored by means of synaptic modifications in the neural circuits which will use this information (McClelland et al., 1995; Karni, 1996). Strong correlations have been found between the activity of individual neurons and behaviour (Newsome et al., 1989; Georgopoulos et al., 1986), and between the activity of individual neurons and sensory stimuli (Desimone, 1991; Perrett et al., 1992; Tanaka, 1996a,b). Hence, cell activity carries information about events. Cells near to the periphery represent low-level features of the sensory input. Cells higher up the hierarchy integrate information and respond to more elaborate 'features' of the input (Crick and Asanuma, 1986).

Both high-level (Tanaka, 1996a) and low-level (Karni, 1996) representations can change with experience. Extended experience leads to increasingly selective receptive fields for certain cells while reducing the total population of active nodes in response to the familiar stimulus (Desimone, 1996). Hence, a smaller active population is associated with improved discrimination performance (Desimone, 1996). In motor learning it is observed that task-specific neural activations develop after practise of a sequence of movements in addition to the initial representations of the elementary movements (Karni et al., 1998). The accompanying improvement in task performance does not generalise, even to other sequences composed of the same elementary movements (Karni et al., 1998). Changes in performance abilities are thus related to changes in the representations which are crucial to the performance of the task (Karni et al., 1995; Karni, 1996). In addition, the cortical regions involved in task performance may

change such that there are fewer active regions once a task is automated than when it is novel (Petersen et al., 1998) (*e.g.*, premotor cortex may be active when learning a skill, but be relatively silent once it has been learnt (Caminiti et al., 1996, p. 174).

Individual cells can have very selective response properties (*e.g.*, in areas IT and STS neurons have been found which are receptive to complex shapes, to faces, and to a single familiar face (Perrett et al., 1992; Tanaka, 1996a,b; Longothetis and Sheinberg, 1996; Perrett, 1996)). However, the majority of cells have broader receptive fields and respond over a range of inputs (Salinas and Abbott, 1995; Abbott, 1994; Mel, 1990a)[6]. The selectivity of cells can vary within a single region, for example, in the primary visual cortex cells are found with simple and complex receptive fields. Both classes of cell receive input from the LGN and hence appear to form part of the same processing stage (Mel et al., 1998; Maffei, 1985), rather than separate stages in which simple cells provide the input to complex ones (contra Hubel, 1988; Ebdon, 1996). Receptive fields are often topologically ordered so that neighbouring neurons have similar preferred inputs (Crick and Asanuma, 1986; Knudsen et al., 1990; Hubel, 1988; Georgopoulos, 1997; Salinas and Abbott, 1995; Swindale, 1996). For example, the primary visual cortex is organised so that it has a retinotopic arrangement, within this map there is further organisation into domains containing neurons with the same ocular dominance preference and within these domains further organisation such that neighbouring neurons have similar orientation preference (Swindale, 1996; Hubel, 1988). Since there is no single terminal region which integrates all information, the visual system generates a distributed representation of visual information (Bartels and Zeki, 1998). In addition, cells within the same region are often active simultaneously, again suggesting that information is coded in a distributed manner.

3.2.3 Development of Mechanisms

Changes in the structure of the cortex underlie behavioural and cognitive development throughout infancy and beyond. Such changes are described as plasticity: "an orderly change in structure or function due to development, experience or injury" (Gregory, 1987). While there is considerable specification of the CNS by phylogenesis (evolution) there is also substantial refinement by ontogenesis (development). This is particularly true of the cerebral cortex which undergoes the most extensive reorganisation. In contrast, subcortical structures are less plastic and their predefined structure underlies innate abilities.

Cells differentiate to form axons and dendrites, however, initial synaptic contacts are imprecise and unstable (Swindale, 1996). During development there is growth of preexisting neurons, and substantial reorganisation of contacts with both the elimination of synapses and the formation of new synapses (Huttenlocher, 1993). In addition, there is elimination of some neural elements through cell death

[6] This result may be biased by the fact that it is very unlikely, given limited testing time, to discover the preferred stimulus of a cell with very high specificity (Földiák and Young, 1995).

(Huttenlocher, 1993).

Cortical development proceeds from the early prenatal period until adulthood, but the principal changes in structure occur during the sensitive, or critical, period of early post-natal development (from 24 weeks gestation until 3 or 4 years, in humans). However, different areas mature at different rates such that peripheral regions (motor and sensor areas) develop early, while central regions (association areas) may not finish maturing until 20 to 30 years of age in humans (Williams, 1983). There thus appears to be a 'wave-of-plasticity' travelling across the cortex which causes different regions to mature at different times (Jacobs, 1999; Shrager and Johnson, 1996). While the wave-of-plasticity might be an innate constraint on the developmental process it might alternatively be an effect of the process of development: the critical period of plasticity for each area might be initiated by the occurrence of patterned activity in the afferent input from lower regions, and terminated by the reducing effect of small changes to the synaptic efficiency as the synaptic strength becomes strong (Munro, 1986; O'Reilly and Johnson, 1994).

Hence, plasticity occurs at different times in different regions. This is just what is required for a constructivist view of cortical development in which more complex abilities are built on lower-level abilities (Quartz, 1999). Regions which mature earlier provide a framework for those which develop later, allowing the later developing neurons to exploit the low-level representations formed in mature neurons in order to develop more complex, higher-level, processing abilities (Jacobs, 1999; Greenough et al., 1993; Shrager and Johnson, 1996; Singer, 1985; Elman et al., 1996). The representations learnt at one level will constrain the subsequent higher-level representations.

At birth the entire neocortex has an almost uniform structure and consistent cellular makeup (except for primary visual cortex (area 17) where there is a greater density of neurons) (O'Leary, 1993; Mountcastle, 1998). Cortical layers are predefined, but cortical areas are not (Elman et al., 1996). Differentiation, or parcellisation, of the cortical sheet into functionally specialised regions results from developmental changes (Jacobs, 1999; Johnson and Vecera, 1996). Modularisation is "the end product of development rather than its starting point" (Elman et al., 1996). Parcellisation results in less widespread information exchange, less interference, and more specialisation of cortical regions (Johnson, 1997; Baron-Cohen, 1996). There is a peripheral to central ordering in timing of regional differentiation (Johnson and Vecera, 1996; Shrager and Johnson, 1996). The functional specialisation of regions reflects this developmental order and is hence influenced by timing factors. It is also strongly influenced by the external input received (Johnson, 1997; Ebdon, 1992, 1996; Greenough et al., 1993; Jacobs, 1999; Johnson and Vecera, 1996; Quartz, 1999). Such input may initially be due to the spontaneous activity of other cells, and only later derive from experience induced activity. However, the mechanisms of adaption are likely to be activity dependent and hence the same for either source of input (Quartz, 1999; Thompson, 1997).

Many studies have been performed which demonstrate that the environment experienced during early development can have fundamental and irreversible effects on the structure of cortical areas (*e.g.*, Cowan and Friedman, 1995; Florence and Kaas, 1995). For example, cats reared in abnormal conditions in which there are no visible horizontal edges will not produce cells with orientation preference to horizontal edges. The behaviour of such animals indicates that they are unable to perceive these features when subsequently placed in an environment containing horizontal edges (Hubel, 1988). Mice reared in an environment enriched with stimuli develop cortical cells with more dendritic branches than mice reared in environments with few stimuli (Quartz and Sejnowski, 1997; Elman et al., 1996; Greenough et al., 1993). Practice of motor actions in humans results in changes to the cortical representation of those actions (Karni et al., 1995, 1998).

As well as modifications to the environment itself it is possible to measure the effect of changes to the way an environment is perceived. For example, the amputation of a limb will cause neighbouring representations to expand into the region originally representing the amputated limb. Other examples are monocular and binocular deprivation experiments on the development of ocular dominance columns in the primary visual cortex (Hubel, 1988; Intrator and Cooper, 1995). With monocular deprivation the ocular dominance of cells moves from the deprived eye towards the open eye. Such plastic changes occur in animals within hours and can be reversed within minutes (Intrator and Cooper, 1995) suggesting that they are due to synaptic changes rather than morphological ones. However, long term monocular deprivation becomes irreversible, and this is assumed to be due to the death of cells which have failed to receive effective stimulation. In binocular deprivation there are no inputs from either eye to stimulate changes and hence little effect on the synaptic organisation. Long term binocular deprivation is reversible (unlike monocular deprivation), possibly because cell deaths are evenly distributed between cells responsive to each eye. Visual experience is not necessary for the formation of retinotopy and ocular dominance columns, however, such experience probably improves the precision of the organisation and can significantly alter the organisation if the stimuli are abnormal (Swindale, 1996). Thus both epigenesis and experience can determine the structure of these maps. For abstract cortical representations activity dependent plastic modifications are likely to play a more important role in their formation. In addition to experimental manipulations to the sensory system, natural conditions can also affect the input that is perceived and hence the structure of cortical areas. For example, the representation of one finger becomes enlarged in braille readers, while the size of the auditory cortex increases in the blind despite auditory stimulation in blinded and sighted individuals being equivalent (Greenough et al., 1993). In the congenitally deaf visual stimulation can cause activation in the auditory cortex (Elman et al., 1996), and similarly the visual cortex is activated by auditory input in the blind (Kujala et al., 2000). Hence, not even the function of primary sensor regions is rigidly predetermined.

Furthermore, experimental manipulations of the afferent pathways, carrying sensory input, also

affect the structure of cortical areas. For example, reducing the thalamic input to a cortical area reduces the extent of that region (Elman et al., 1996). Rerouting inputs intended for the primary visual cortex onto the auditory cortex causes what is normally the auditory cortex to develop response patterns and receptive fields typical of the visual cortex (Sur, 1989; O'Leary, 1993; Johnson, 1993; Johnson et al., 1998; Johnson, 1997; Elman et al., 1996).

Similarly, manipulation of the cortex itself reveals that a cortical region can develop the structure of a different region when it receives afferents intended for that region. The removal of a section of cortex and its transplantation within another region of the cortex will result in the transplanted section taking on the structural appearance of cortex typical for the site it was transplanted to rather than the site it was taken from (O'Leary, 1993; Johnson et al., 1998; Johnson, 1998, 1997). Such transplanted sections also take on the input and output characteristics of their new location (Elman et al., 1996). Focal damage to cortical regions in early infancy causes other areas (possibly at considerable distance from the damaged region) to take up the function of the damaged region (Johnson, 1997, 1998; Elman et al., 1996), supporting the view that the functionality of regions is not prespecified (Johnson et al., 1998). Hence, regions in the immature cortex appear to be able to take on the appearance and functionality of others and have equal potential for representing different domains (Sur, 1989; O'Leary, 1993; Johnson, 1997, 1998).

The evidence presented above suggests that the detailed structure of a cortical region reflects the input received and hence changes to the environment, to the sensory-motor system, to the afferent pathways, and to the cortex itself all affect the neural representations developed. Furthermore, some evidence supports the view that the same developmental mechanisms operate across all cortical regions. Hence, from a developmental perspective the cortex is even more uniform than it is from a structural perspective. Understanding the organisational process in one area should give understanding of all other areas and data from different areas and domains can all be used to constrain a single model.

Variations in the structure between areas of the mature cortex may thus be due to differences in the information being processed rather than intrinsic differences between regions. The differentiation of the cortex into regions of functional specialisation thus results from the developmental process and is not predefined, and the structure of the input is essential to the structure of the module (Karmiloff-Smith, 1994). There are no domain specific mechanisms rather development relies on the same mechanisms being applied in all domains (Karmiloff-Smith, 1994).

There is considerable evidence that the same capacity for activity-dependent changes in synaptic effectiveness persists into adulthood (Mountcastle, 1998). Hence, the same mechanisms may be responsible for learning as for development, but generally with less wide ranging architectural changes resulting from learning (Singer, 1995, 1985; Bear, 1985). However, learning does cause measurable effects on cortical representations as a result of changes in the environment (Recanzone et al., 1993)

or as a result of learning new skills (Karni et al., 1998; Nudo et al., 1996). However, the scope for more wide ranging structural changes may be more limited, so that while the human child can recover higher cognitive functions (*e.g.*, language, spatial cognition) following focal damage to a cortical region, similar injury in the adult results in irreversible deficits (Elman et al., 1996). While capacities for structural change may be more limited in adults (Bear, 1985) the adult cortex does retain the capacity for dendritic growth and synaptogenesis (Greenough et al., 1993; Gilbert et al., 1996) and considerable structural reorganisation can occur in adulthood especially in response to changes in experience caused by injury (Das, 1997; Bear, 1985; Kaas et al., 1997; Kaas, 1991; Florence and Kaas, 1995; Greenough et al., 1993; Sanes and Donoghue, 1997). These structural reorganisations appear similar to the plastic changes which occur in response to similar manipulations of the input during development. Hence, the same mechanisms underly plasticity due to pathological changes as underly plasticity due to normal development (Elman et al., 1996).

It is not only late maturing regions which retain this capacity. Plastic changes can occur at all levels of the adult somatosensory system (Florence and Kaas, 1995), and skills which are dependent on low-level, stimulus dependent processes can show significant improvements in adults after training, showing that even low-level representations remain plastic (Karni, 1996; Karni and Bertini, 1997; Karni et al., 1995; Gilbert et al., 1996).

3.3 Specification of Intelligence

3.3.1 Methods of Specification

Appraisal of the role of development has tended to concern the relative influence of genetic and environmental factors, or 'nature' and 'nurture', in the developmental process.

3.3.1.1 Nativism

This interpretation suggests that genetic factors are predominant and hence that the environment at most provides triggers for the maturation of prespecified capacities and possibly provides some parameter adjustment for these capacities (Rokers, 1998). The role of development is then simply to allow maturation of genetically predefined abilities and knowledge. This point-of-view has its roots in rationalism, the philosophical position that knowledge comes prior to experience (Honderich, 1995; Rokers, 1998). This extreme has difficulty explaining how it is possible to ever know or learn anything new since all knowledge is innately predefined.

3.3.1.2 Empiricism

This interpretation suggests that environmental factors are predominant and hence that experience of the animal and its interaction with the environment are primary in the formation of new capacities (Rokers, 1998). The role of development is then to allow the *tabula rasa* learning of abilities and knowledge. This point-of-view has its roots in empiricism, the philosophical position that all knowledge comes from sensation *i.e.*, through experience (Honderich, 1995; Rokers, 1998). This extreme has difficulty explaining how it is possible to ever know or learn anything new since unless knowledge can already be represented somehow within the agent there can be no possibility of learning it.

3.3.1.3 Interactivism

Neither extreme position in the nature-nurture debate seems tenable. The correct theory will combine aspects of both (Karmiloff-Smith, 1994, 1993): "the neocortex acquires knowledge of the world by nature as well as by nurture, but these methods work towards the same ends rather than being ... mutually exclusive alternatives" (Barlow, 1994). Development thus involves the interaction between genetically controlled maturation processes and the environment: "we propose that the developing individual is an emergent product of a dynamic interaction between nature and nurture. The emergent individual is to be found neither in the environment nor in the genetic start state, but in the rich dynamics between them" (Plunkett et al., 1997). The interactions between genes and their environment (and between one gene and another) mean that there is not a simple linear relationship between the genotype and the phenotype (Elman et al., 1996).

That such an interactivist interpretation is necessary is obvious. For nativism the environment must at least provide conditions to allow the genetic code to be expressed (even a computer program relies on the correct context – the interpreter, the operating system, the hardware – for it to work (Clark, 1998)); for empiricism the genetic code must at least define the learning mechanism and perceptual-motor abilities that could allow learning from experience. However, this still leaves open the question of the relative importance of environmental and genetic factors in this interaction. The nature-nurture debate thus becomes one about how much knowledge is innate and how much is learnt. The details of the connectivity of individual synapses could not be explicitly predefined in the genetic code, since there is insufficient information in DNA (von der Malsburg, 1995) (the human genotype contains approximately 10^9 bits of information, the human brain contains roughly 10^{14} synapses). It must thus be the case that the DNA specifies a mechanism through which synapses are produced. It is simply the degree to which this mechanism is influenced by the environment that is under debate. Hence, issues such as the extent of the initial repertoire of abilities at birth and the timing of when particular capacities develop have become central. However, attempting to make an arbitrary distinction between abilities most strongly influenced by one factor or the other is of limited value in understanding the developmental process: in

attempting to understand how something develops it is not possible to single out privileged factors since all play a causal role (Clark, 1998) (in contrast, in attempting to understand why something develops, rather than how, it may be appropriate to single out individual genes as the principal cause (Clark, 1998)).

Development, as an interactive process, can be distinguished from the predominantly genetic process of maturation as well as from the predominantly experience-dependent process of learning: development is more than just growth, and more than just parameter adjustment. Hence, for interactivism, development is a separate stage that cannot be ignored in any theory of intelligence.

Structure Definition: maturation.

Phylogenesis provides specification of the innate repertoire of capacities, structural morphology and the mechanisms for subsequent ontogenesis.

Structure Adjustment: development.

Development provides expansion of the behavioural repertoire.

Parameter Adjustment: learning.

Learning provides tuning of the behavioural repertoire (Beer et al., 1991).

Evolution can be considered to be a meta-level of adaption to development, which is itself at a meta-level to learning. The distinctions between these levels are fairly arbitrary. Distinguishing development from learning is particularly difficult. Similar neural mechanisms appear to underly both processes (and the model that is implemented in the following chapters of this thesis relies entirely on learning mechanisms). The distinction is thus likely to be a qualitative one rather than a firm quantitative one. At the behavioural and cognitive level development can be distinguished from learning since it has a longer time span and results in progressive and qualitative changes in ability (Moshman, 1996). At the neural level development is characterised as a process requiring changes in the neural architecture itself (sections 3.4.1 and 3.4.2, Quartz and Sejnowski, 1997), while learning is a process of synaptic modification within a fixed architecture. Hence, development results in changes to the representational space while learning changes the representations within a space (*cf.*, section 2.3.3.1, Quartz and Sejnowski, 1997). The model presented here also relies on changes in neural architecture but through new neural regions becoming involved in learning, rather than through selectionism or constructionism within a single region. Learning in one region alters what can be learnt subsequently by other regions by changing the representational space in which subsequent learning takes place. Development is thus seen as learning which relies on changes in the representational space caused by previous learning. Such changes can mean not only that new skills are learnt but also that old skills become performed in new ways. Development is further characterised, and distinguished from maturation, in section 3.3.2.2.

It seems likely that early in life, phylogenetic processes are most dominant, but as the agent becomes

more complex ontogenetic processes become more and more important (Vaario, 1994), since it is unlikely that late cognitive development could be genetically preprogrammed (Moshman, 1996). Although there is considerable specification of the agent's structure and innate abilities provided by phylogenesis, there is also considerable adaption during the agent's life time, and acquired abilities develop through ontogenesis (Schnepf, 1991). Phylogenesis must define the anatomy of the animal (for it to exist) and this will define how the agent can interact with and perceive the external environment. The agent's structure thus places constraints on its abilities. Evolution provides constraints on cortical development but few explicit rules for cortically controlled behaviour: preadaption without rigid predetermination (Rutkowska, 1996). Evolution, however, must also define the mechanisms for cortical development and hence, in a sense, also specifies all subsequent ontogenesis. However, in this view evolution is seen as providing innate architectures rather than innate representations as would be required for nativism (Elman et al., 1996; Plunkett et al., 1997). Representational nativism is unlikely in a cortex which is so plastic (Elman et al., 1996) but total representational freedom is equally unlikely in a cortex which is so uniformly structured.

3.3.2 Advantages of Development

3.3.2.1 Advantages Over Predefinition

For predictable environments there is an increased chance of survival if behaviour is innate, however, in unpredictable environments it is biologically advantageous for behaviours to be adaptive to experience (Encyclopaedia Britannica Online, 1997). There is a trade-off between robustness of behaviour and the time required to develop it. In addition, to generate more complex abilities nature seems to have found it necessary to rely on the developmental process, rather than directly evolving extremely complex structures (Pallbo, 1994). "A general evolutionary trend exists for more and more behaviours to be modified by environmental stimuli as the phylogenetic scale ascends (*i.e.*, as the animal becomes more complex and "advanced"). This tendency has reached its extreme expression in vertebrates, particularly mammals, and especially in humans" (Encyclopaedia Britannica Online, 1997). Development may thus be necessary to make tractable the task of forming complex abilities (see section 3.3.2.2).

Biological agents thus have the ability to perform complex tasks in complex environments, but for higher animals, only the potential for such abilities exists at birth: the newborn is incapable of performing even simple actions, such as walking, without first undergoing a period of development. A greater reliance on development results in an agent being better able to cope with abnormal conditions and hence being less brittle to environmental change or damage, being able to cope with a larger variety of environmental conditions (Elman et al., 1996) and allows motor responses to be applied to new tasks, and new behaviours to develop. The manner in which nature produces intelligent behaviour thus

has several properties desirable in engineered solutions in contrast with the problems caused by using prespecification in robotics (section 2.3.2).

1. **Tractability.**

   Evolution relies on development, whenever possible, to avoid the need for detailed prespecification and to make tractable the task of creating complex abilities.

2. **Non-Specificity.**

   Similar mechanisms account for the development of various capacities across individuals and across species.

3. **Robustness.**

   The brain adapts to the environment and learns to cope with novel situations autonomously.

4. **Adaptability.**

   Biological agents can expand their behavioural abilities, and continue to adapt to the environment and learn from experience.

3.3.2.2 Role of Constraints

In contrast to nativism, in which development is seen as guided towards an innately defined end point, interactivism requires that development be guided by a predefined starting point that provides constraints and biases on subsequent development. Both evolution and the environment can be thought of as providing constraints on the developmental process (Johnson, 1993) (in contrast, for nativism there would be only endogenous constraints while for empiricism there would be only exogenous constraints (Elman et al., 1996)). Sources of constraint come from (Johnson et al., 1998; Johnson, 1993, 1997):

- the anatomy of the body and sensory-motor system;
- the architecture of the brain (de Schonen and Mancini, 1995);
- the temporal dynamics of development (the wave-of-plasticity) (Quinlan, 1998);
- innate, subcortically controlled behaviour (de Schonen and Mancini, 1995; Johnson, 1997, 1998);
- spontaneous neural activity (Eglen, 1997; Johnson, 1998; Shatz, 1996; Thompson, 1997);
- the environment (Quinlan, 1998).

The classification of a particular constraint as endogenous or exogenous will depend on the definition of the external environment for that particular process (*e.g.*, should the spontaneous neural activity that gives rise to retinal waves be considered part of the animal's genetic make-up or part of the environmental input to the visual cortex?) (Elman et al., 1996). Constraints progressively restrict fate during

development (Johnson, 1998). As the brain and body develop, these constraints change, which in turn alters what can be subsequently learnt. Developmental theories thus need to characterise a non-stationary learning process (Quartz and Sejnowski, 1997). Initial synaptic plasticity will be due predominantly to the input received. The synaptic structures thus formed will, in turn, affect the neuronal activity, and hence subsequent plasticity (Pallbo, 1994). The initial plasticity will happen in response to the most ubiquitous features of the input to the brain, subsequent plasticity is constrained by the results of previous experience and so will extract increasingly more detailed representations (Pallbo, 1994). Both the brain and peripheral sensory-motor systems develop together (Atkinson and Braddick, 1989; Slater, 1989). Hence, during development there is a gradually increase in complexity in: the environment perceivable through the senses; the cognitive architecture; and the motor skills that can be performed.

Some pre-existing abilities are required to under-pin the development of more complex abilities (the agent must be viable and already able to operate in the environment before it can adapt its behaviour (Steels, 1993)). Innate abilities are provided through subcortical mechanisms. Hence, while there is a lack of innate representation in the cortex this is not true of the brain as a whole (Johnson, 1997).

Simple abilities can provide bias in order to make learning complex behaviour tractable. In a constructivist developmental process, high-level cognitive abilities are developed through a progressive increase in competence achieved by using simpler abilities as the basis from which to learn more complex abilities (Elman et al., 1996; Arbib, Conklin and Hill, 1987). By first learning the most salient features of the input, or a sub-set of a complex problem, it becomes possible to subsequently learn to process more subtle features, or to deal with the full problem, when such a task could not be solved in a single step (McClelland and Plaut, 1993; Elman, 1993; Elman et al., 1996). Simple abilities thus provide constraints to guide the learning of more complex abilities (*cf.*, the bias-variance problem; section 2.3.3.1). Hence, rather than being a limitation on performance, early developmental restrictions may be a prerequisite to learning complex tasks (Jacobs, 1999) and limited abilities are not deficiencies to be overcome but constraints which facilitate development (Turkewitz and Kenny, 1993). Cortical limitations (*e.g.*, due to timing of maturation of memory or attentional systems) as well as physical limitations (*e.g.*, due to timing of the maturation of the body) facilitate development by reducing the complexity of the initial task to be learnt (Elman et al., 1996). Early behavioural skills have sensory effects and hence the range of possible actions constrains experience. As motor ability increases this provides a basis for learning more complicated behaviour. Early sensory limitations may be important in enabling sensory systems to organise, by providing an orderly world to the infant (*e.g.*, limited depth of field allows learning of relative sizes and thence size constancy) (Turkewitz and Kenny, 1993). Extrinsic input can be biased by subcortical attention biases and predispositions (Johnson, 1997; Karmiloff-Smith, 1993; Greenough et al., 1993). For example, orientation to face-like shapes is an innate, subcortically controlled reflex which selectively biases the input to the cortex during development with face-like stimuli so that the

cortex will develop a module for face processing that is not present at birth (de Haan et al., 1998). Only later does recognition of species-specific faces or details of facial features develop. Similarly, imprinting uses an orienting system to bias the input to a learning system (Johnson, 1992, 1999; O'Reilly and Johnson, 1994). As the cortex matures it must be able to influence subcortical behaviour (*e.g.*, to inhibit orientation towards faces, or gate the oculomotor reflex performed by the superior colliculus (Atkinson and Braddick, 1989)).

Evolution provides sufficiently strong constraints on development that there is significant uniformity across individuals of the same species and in the developmental progression across individuals. These similarities are not support for nativism but are due to the similarity of innate predispositions, of anatomical structure, and to the similarity of the environment (Karmiloff-Smith, 1994). Two categories of plasticity can, thus, be defined: experience-expectant plasticity (due to the species-typical environment (Johnson, 1993) *i.e.*, those conditions which all members of the species are likely to be exposed to) and experience-dependent plasticity (due to the individual-specific environment (Johnson, 1993) *i.e.*, the particular experience of the individual, including abnormal experiences) (Greenough et al., 1993). Outcomes dependent on interactions in the species-typical environment are the same for all individuals and are hence most strongly influenced by evolution[7]. Outcomes dependent on interactions in the individual-specific external environment are learnt (Elman et al., 1996).

There is a strong predisposition for the cortex to develop the standard pattern of parcellisation despite the significant plasticity that is possible. Hence, there must be some strong constraints on regional development (Johnson et al., 1998). Rough connectivity provides strong predispositions (Greenough et al., 1993), and the position of a region of cortex is decisive in determining the area-specific properties it develops under experience-expectant conditions (O'Leary, 1993). The timing of development also affects the specialisation of cortical regions (Elman et al., 1996). All (healthy) individuals will have similar body morphologies and the projections of afferent connections from the sensors into the brain and of efferent projections out of the brain will be similar. Also, the rough topology of axonal growth is not activity-dependent and thus provides a constraint upon subsequent activity-dependent refinement (Bear, 1985). The large-scale architecture of the cortex is thus relatively insensitive to experience (given species-typical conditions) while the small-scale architecture is still sensitive (Johnson et al., 1998). Hence, for a model of normal cortical development it is the small-scale plasticity that is of principal interest, not parcellisation. Under normal conditions development changes the small-scale structure within a predefined architecture and refines the predefined interconnectivity between regions. Under abnormal conditions both small and large-scale architectures may change and a different parcellisation

[7] Some constraints may be so strong that they can be said to determine the outcome of development. Since no traits are coded in the gene such that they are not influenced by some level of interaction, innate behaviour could be defined as that which is more or less inevitable or dependent only on molecular or cellular interactions but not on external information (Elman et al., 1996).

results (Johnson, 1997).

3.4 Models of Development in Biological Systems

3.4.1 Neural Selectionism

Applying Darwinian ideas of natural selection to the nervous system has become very popular (Purves et al., 1996). It is proposed that competition between neural structures results in only those that prove useful surviving, or being selected. Evidence used to support this idea includes the pruning of excessive, low-precision connections during the critical period (Singer, 1985), cell death, and variations in neural structure between individuals (Purves et al., 1996).

However, in order for there to be competition between neural structures these structures must first be formed. Theories differ in the way in which they propose that the competing structures are created. Either;

- they are produced randomly (Pallbo, 1994; Changeux and Dehaene, 1993), or
- they are predefined (Edelman, 1987).

The first mechanism is an optimisation process and uses unguided construction together with elimination guided by experience. The second mechanism requires the formation of the initial neural structures to be genetically predefined and hence only causes regressive changes: the entire repertoire of possible neural structures must exists before-hand and is selected between (Crick, 1989).

3.4.2 Neural Constructionism

Contrary to the requirements for selectionism there is actually an increase in synaptic numbers (due to growth in cell arborisations) until sexual maturity (Purves et al., 1996; Huttenlocher, 1993) (only after which is there is a decline). Progressive changes would thus appear to be more important than regressive ones during neural development (Quinlan, 1998; Purves et al., 1996; Quartz, 1999). Neural constructionist theories thus emphasise that the formation of neural structure is guided by experience, in contrast to selectionist theories in which the elimination of neural structure is guided by experience (Greenough et al., 1993). For example, Quartz and Sejnowski (1997) propose that dendritic growth is the mechanism for constructionist learning. "The representational features of the cortex are built from the dynamic interaction between neural growth and environmentally derived neural activity" (Quartz and Sejnowski, 1997).

However, it seems likely that neural development makes use of both progressive and regressive changes in synaptic structure. The initial, rough connectivity between neurons provides a strong bias

for development. Within this large-scale, innately defined structure there will be activity-dependent formation and elimination of synapses. Both synaptic sprouting and pruning is observed, and the mechanisms underlying these processes are considered to be similar to the processes responsible for synaptic strengthening and weakening (Miller, 1998; Elman et al., 1996). This suggests that progressive and regressive changes in synaptic structure can be approximated by a single algorithm that provides for synaptic strengthening and weakening. Most learning rules used in artificial neural network models provide for both increasing and decreasing the synaptic strength, and hence, most models assume that both constructive and selectionist processes are necessary.

3.4.3 Genetic Epistemology

Piaget's explanation of development is strongly interactivist. He proposes that there is a two-way process of mutual interaction between schemas and the environment which drives cognitive development. He proposes that this is underlaid by two fundamental processes (Drescher, 1991; Gregory, 1987).

Assimilation. The activation of schemas by stimuli.

Accommodation. The adaption of schemas by stimuli.

The combination of which is equilibration. These processes remain rather vague (Bates and Elman, 1993), however, connectionist modellers have equated assimilation with the activation of a neural network and accommodation with learning in a neural network (Mareschal and Thomas, 2000). These continuous underlying mechanisms are proposed to give rise to the stage-like developmental process described in section 3.1.3. Stage-like transitions have also been found to occur in connectionist models in which the input distribution and learning algorithm remain constant: a gradual and continuous process can thus give rise to stage-like behaviour, due to quantitative changes in the weights or qualitative changes in the network structure (Elman et al., 1996; McClelland and Plunkett, 1995; Plunkett et al., 1997; Shultz, 1991).

One source of criticism of Piaget is the proposal that stage transitions occur simultaneously across all domains (Shultz, 1991). It seems more likely that capacities for different domains of behaviour develop at different rates, but due to the mutual interaction of behaviours the maturing of capacities in one domain will affect the development of those in other domains. In this way domain-general mechanisms, operating independently within individual domains, can still account for development (Karmiloff-Smith, 1994). Piaget would also appear to be wrong in claiming too few predispositions (being too empiricist). Recent evidence shows that skills are in place earlier than Piaget describes (Slater and Bremner, 1989) (*e.g.*, the coupling of some perceptions to actions in the very young (Bertenthal, 1996)). These precocious abilities may provide evidence either for faster skill acquisition than originally thought, or for the number of innate predispositions having been underestimated.

Piaget provides a convincing argument for the nature, and origin, of knowledge (his theory of genetic epistemology through constructivist development) which emphasises the need for active participation of the child and its interaction with the environment.

3.4.4 Representational Redescription

Karmiloff-Smith (1994) proposes that development results from a process of three recurrent phases occurring in each domain. The first phase is data driven and culminates in successful performance in that domain (behavioural mastery). The second phase involves the recoding of the procedural representations, formed from phase one, into declarative representations. This redescription process leads to increased performance errors. In the third phase there is reconciliation between redescriptions and data. Representational Redescription "is the hypothesised process by which information already IN a cognitive system becomes progressively explicit knowledge TO that system. Development thus involves two complementary processes of progressive modularisation and rendering explicit" (Karmiloff-Smith, 1994). Development is seen as a process of modularisation, which make knowledge less accessible, and explicitation which makes knowledge more accessible (to other parts of the cognitive system). Karmiloff-Smith (1994) implies that there is a prespecified format of representation that the redescription process is aiming to produce to allow information to be more explicit and transmitted between domains. However, it seems unlikely that the most central 'language of thought' is prespecified but not the peripheral ones. Similarly, Piaget considers the integration of schemas from different modalities as one problem to be overcome by the developmental process. However, this is only the case if it is assumed that representations are stored in a modality-specific format (Bertenthal, 1996). The uniformity of the cortex suggest that there are no such domain-specific formats. In addition, the parcellisation of the cortex during development (section 3.2.3) suggests that initial inter-sensory connectivity is differentiated, rather than sensory domains being initially independent and development leading to integration (Baron-Cohen, 1996).

3.4.5 Self-Organising Maps

Many algorithms have been proposed to model the self-organisation of cortical maps (see Swindale, 1996; Erwin et al., 1995, for reviews of models of the visual cortex). Most of these models are of ocular dominance or orientation preference (*e.g.*, von der Malsburg, 1973; Goodhill and Willshaw, 1990, 1994; Goodhill, 1993; Erwin and Miller, 1996; Marshall and Kalarickal, 1996; Sirosh and Miikkulainen, 1995; Sirosh et al., 1996; Sirosh and Miikkulainen, 1994; Miikkulainen et al., 1997). These algorithms have considerable similarity (Swindale, 1996). The cortex is represented as a single sheet of neurons in which each node receives afferent excitation from a receptive field (RF). The RFs are refined by a learning algorithm to generate a topologically organised representation of the input space. In many

cases the initial receptive fields are predefined, however, the rough initial topography can be generated by using a model of axon growth controlled by chemical markers followed by refinement by activity-dependent synaptic modification (Hiller, 1993). Activity-dependent learning relies upon: patterned afferent activity; pseudo-Hebbian synaptic modification; normalisation of the total synaptic strength of each neuron; and lateral connections between neurons in the cortical sheet which cause excitation over a short range and increasing inhibition at greater distances. It is obvious that this form of lateral connectivity will encourage nearby neurons to be active for similar input patterns, while competition between more distant neurons will result in them coming to represent dissimilar inputs. In this way such a network can come to form a topologically organised map.

While these models may provide some insight into the processes underlying the organisation of individual cortical regions, they do not model cognitive or behavioural development.

3.4.6 Supervised Learning Models

Several processes in cognitive development have been modelled using supervised learning algorithms (*e.g.*, Shultz, 1991; Elman, 1993; Elman et al., 1996; Plunkett et al., 1997). However, supervised learning requires that the algorithm is supplied with the required outputs that are to be generated in response to each set of inputs in the training data. This makes such models implausible since if the required internal representations are known then there is no need to learn them (Marcus, 1998). The internal representations must be innately predefined and hence there can be no development occurring.

3.5 Summary

Development is a process through which animals expand, and increase the complexity of, their cognitive and behavioural capacities. Such intellectual development is predominantly a result of changes which occur in the cerebral cortex. Experience plays a significant role in shaping the development of these abilities and the underlying changes to the cortical architecture.

The neocortex is uniform with regard to both its structure and its developmental mechanisms. Neurons in the cortex learn to respond to events in the world. Such representations are learnt by recording re-occurring structures in the environment. The neocortex is composed of anatomically and functionally distinct regions or areas. Regions along particular pathways form a loose hierarchy in which information propagates from peripheral areas to more central ones. Peripheral regions mature earliest and hence later developing neurons, in more central regions, can exploit those representations which have been formed, in more peripheral regions, in order to develop more complex, higher-level, representations. Development is thus a constructivist process which is reflected in the progressive developmental process observed at the behavioural level: simpler cognitive and behavioural abilities provide a basis from

which to learn more complex and abilities, and so on hierarchically.

The following chapter presents the connectionist model of development which has been implemented. In this model neurons learn to represent re-occurring events in the world, and by using a hierarchy of identical neural network modules more complex representations can be learnt in higher-level networks based on simpler representations learnt in lower-level networks.

Chapter 4

IMPLEMENTATION

The previous chapters have motivated the use of a developmental process for the formation of complex cognitive and behavioural abilities. This section describes the model of development that has been implemented. It is the development of the cerebral cortex which underlies development in animals. While the gross anatomy of the cortex is well documented the complexity of its detailed physiology means that experimental neuroscience provides only a partial understanding of its functioning. Rather than try to develop a model of development based on the known neurological facts the model described here has been inspired by neuroscience and developmental psychology (as discussed in chapter 3) and by consideration of the requirements for mechanisms which could provide cognitive and behavioural development in an abstract artificial agent (as discussed in chapter 2). This project has thus taken an integrative approach in which all these considerations have been used to inspire and constrain the model. The result is a model which is broadly consistent with neurology without being a faithful model of the brain, is described in terms of psychological behaviour without modelling any particular experimental data, and could be a step towards providing development in robots without being of immediate practical value.

Since the aim is to produce a model that is consistent with data from, and makes predictions at, both the cognitive and neural levels the implementation must be at an appropriate, intermediate, level of abstraction. To this end a standard connectionist style neural network has been used. An artificial neural network consists of a group of separate processing units which provide a simplified and generalised mathematical model of a biological neuron. Such networks thus allow consideration of the computational properties of neurons rather than the detailed physiology of cells, and their functioning can be given a direct interpretation at the cognitive and behavioural level. Such models are abstractions which trade-off faithfulness to low-level biological detail for improved functional performance. For example, many highly influential models of lateral inhibition are physiologically implausible, requiring excitatory

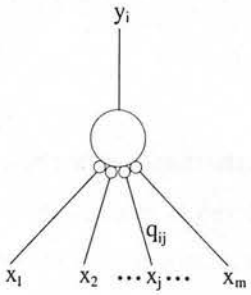


Figure 4.1: A model neuron. The node body is shown as a large circle. It receives inputs $(x_1, x_2, \ldots, x_j, \ldots, x_m)$ via excitatory synapses shown as small open circles. Synapses have an associated weight (q_{ij}). The activation of the node is output as value y_i.

cells to directly inhibit other excitatory cells (*e.g.*, von der Malsburg, 1973; Sirosh and Miikkulainen, 1996b; Földiák, 1990; Marshall, 1995a; Swindale, 1996). Far from being a deficiency such abstraction is the very thing that provides models with their explanatory and predictive power. Thus the biologically plausibility of a model should not be equated solely with its degree of correspondence with low-level details (if this were the case then all models pitched above the level of sub-atomic particles and fundamental physical forces would be open to attack for implausibility). Correspondence at a higher-level (in this case with behavioural, cognitive and computational requirements) is equally valid. Since both high-level and low-level consideration have been used to inspire and constrain the model presented here there is no claim that it is anatomically or physiologically justified in all (if any) of its details, but that it is nonetheless biologically relevant.

The behaviour of a model neuron (figure 4.1) is determined by calculating its activation; a computation performed in two stages. Firstly a 'combination function', f_2, derives a single value from the pre-synaptic activation values, x_j (the 'inputs'), and the synaptic weight values, q_{ij}. This value is then passed through an 'transfer function', f_1, to derive the activation of the node, y_i (the 'output').

$$y_i = f_1\left(\sum_{j=1}^{m} f_2\left(q_{ij}, x_j\right)\right). \tag{4.1}$$

Collectively the process of calculating the output from the inputs and the synaptic weights will be referred to as the 'activation function'. Standard combination functions include a simple linear sum of inputs modulated by the corresponding synaptic weights $(\sum_{j=1}^{m} q_{ij}x_j)$, and the Euclidean distance between the vector of input values and vector of weights $(\sum_{j=1}^{m} (q_{ij} - x_j)^2)$. Standard transfer functions include the identity function and the sigmoid function. In addition, the activation of each node may also be affected by the output of other nodes in the network through lateral or recurrent processes.

The output, or activation, of the nodes in a network constitutes a response to the current input to the network. This response can be considered to be a representation, or a perception, of the input stimulus. The response of a neuron can be modified through a learning procedure which adjusts the synaptic

weights with the aim of improving the representation of the input data. This will modify the behaviour of the model neuron and can thus be used to provide learning or development. A simple, biologically plausible, learning rule which uses information locally available at each individual synapse adjusts each weight as a function, f_3, of the pre- and post-synaptic activity at that synapse:

$$\Delta q_{ij} = \beta f_3(x_j, y_i). \tag{4.2}$$

Such synaptic plasticity is thus activity-dependent. Standard learning rules may be either supervised (in which case the required post-synaptic activation is specified for each input pattern), or unsupervised (in which case only the post-synaptic activity, determined by the activation function, is available). Other learning procedures which use non-locally available information or which optimise an objective function for the network as a whole are also in common use.

Each input thus has both short-term and long-term effects on the activation of a neuron, through activation and learning respectively[1]. The inputs from which a node receives (strong) synaptic weights constitute the receptive field (RF) of that node. These inputs are those from which the node receives excitation and will also be described as the preferred inputs, or the represented inputs, of the node. The RFs of all nodes in a network are refined by the learning algorithm to generate a representation of the input space. The learning algorithm forms RFs over an extended period of exposure to input data. The input of a single stimulus, and the subsequent calculation of the output activation and adjustment of the synaptic weights, is termed an 'iteration' or 'training cycle'.

This chapter presents a novel neural network algorithm. This model uses two separate versions of the combination function: a linear and a nonlinear version (section 4.4). A simple variable threshold is used for the transfer function. Lateral connections between the nodes in the network also affect node activations, as does past activity and noise (section 4.3)[2]. Unsupervised learning is used (section 4.2). However, a second source of inputs (described as apical whereas the standard source of input is described as basal) can be used to modulate the learning (section 4.2.4). In addition, two variations of the learning rule are allowed: one (afferent) in which learning occurs after the outputs of the nodes have been updated, and another (efferent) in which learning occurs before the outputs of the nodes are updated (section 4.2.3). The algorithm is stated, for ease of reference, in tables 4.4, 4.5, 4.6, and 4.7. At this point only table 4.4, which defines the symbols to be used in equations, is relevant to the reader. In many applications the algorithm can be simplified to that given in table 4.8. In this formulation it can

[1] Potentially explicit knowledge, stored in the synaptic weights, has a causal role in processing knowledge explicitly represented by the node activations. In contrast, a symbol system can only make use of potential knowledge after recall to explicit memory, requiring search since the relevance of such knowledge cannot be known until after recall (O'Brien and Opie, 1999).

[2] Since a nonlinear threshold is used as the transfer function, and additionally the lateral interaction makes use of a nonlinear selection process, nodes are always nonlinear, however, the term nonlinear will be reserved for nodes using the nonlinear form of the combination function and nodes using a linear combination function will be referred to as linear.

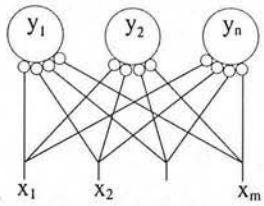


Figure 4.2: A model neural network. A single layer of nodes all receive excitatory connections from an array of inputs. The activation of the nodes in the network constitutes a representation of the current input. Lateral interaction between the nodes affects this output representation and due to learning affects the receptive fields.

be seen to be similar to a standard competitive learning algorithm. It may thus be easier to consider this formulation initially and to think of the other mechanisms as embellishments to this simpler algorithm. The algorithm is described in more detail, and qualitatively justified, in the following sections of this chapter[3]. Quantitative examples of its performance are given in subsequent chapters.

In common with most unsupervised learning algorithms the network consists of a single sheet of neurons (figure 4.2), in which all the nodes receive excitatory connections from an array of input sources. An individual network will have a particular functional specialisation and hence, in analogy with the cortex, will be described as a region. Unlike the cortex, in which there is parcellisation in to regions in response to experience, regions are predefined in this model. The algorithm learns an encoding appropriate to the structure of the input data received, and works with the same formats of encoding for both inputs to and outputs from the network. This enables the transfer and recoding of information between separate networks to be modelled by using the output of one region as (part of) the input to other regions. There is thus uniformity of coding across the model. This allows a modular 'assembly' of neural networks to be used for learning more complex behaviour. The algorithm enables each neural network to self-organise into a representation of its input at the same time as the relationship between these representations is found. Unlike previous methods learning is achieved without a separate training phase. There is thus uniformity of the algorithm throughout time, although in some cases the time at which learning is initiated varies between regions. All model regions, in any assembly, will consist of this same neural network. There is thus also uniformity of the algorithm throughout the assembly of networks.

Since the same learning algorithm is to be used in all the regions of an assembly it is important that it be robust to changes that may occur between regions (such as the number of nodes and the number of inputs). Parameter tweaking also needs to be avoided since changing the parameters in one region to improve its performance will affect the input supplied by that region to other regions and hence affect their performance, in which case finding suitable parameters for all networks would become a difficult optimisation problem to solve. The algorithm presented in this thesis overcomes these problems and is

[3] Only numbered equations in the main text are directly relevant to the derivation of the equations used in the algorithm.

used, with identical parameter values, in all regions in an assembly. As well as robustness across regions it is also desirable for the algorithm to be robust across applications, the aim being to develop a generally applicable model rather than a large variety of application specific models illustrating different aspects of development. The same algorithm is used throughout this thesis, and all results which are shown have been generated using identical parameter values (as given in table 4.4), except for four parameters ($\sigma_I, \sigma_E, \lambda_x$, and κ) which have variable values[4]. The algorithm has thus proven to be robust by not requiring parameter adjustment for successful performance on different problems. The robustness of the algorithm is further demonstrated by the fact that (random) changes in the parameter values have little affect on the performance (see chapter 5).

The algorithm is part of the innate architecture of an assembly. Additional innate aspects include the pre-specification of the regions forming an assembly, the connectivity between regions, and connectivity between the regions and the input[5]. These all provide predefined constraints on the representations formed in each region. Some constraints on the timing of events are also imposed to ensure synchronisation between regions (section 4.2.3).

4.1 Coding Requirements

The pattern of activity of the nodes in a neural network constitutes a representation of the input to that network. In unsupervised learning, unlike supervised learning, what node activations constitute a good representation of the current input is not explicitly defined by the training data. Instead, the required representation is implicitly defined by the input data and the learning algorithm. Thus many different algorithms have been proposed (see Barlow, 1989; Becker and Plumbley, 1996, for reviews) which impose various, application independent, measures of 'goodness' on the representation formed; such as information preservation, minimum redundancy, minimum entropy, density estimation over priors, maximum likelihood parameter estimation, topology preservation *etc.* Such algorithms may be implemented using local learning rules or via the optimisation of an objective function (Becker, 1995). By modifying the learning rules and activation function, or by modifying the objective function, the same neural network architecture may be used to find representations that preserve different measures of 'goodness' (*e.g.*, (Oja, 1995), *cf.*, (Földiák, 1990) and (Földiák, 1989), *cf.*, (Fyfe, 1997) and (Harpur and Prager, 1996), *cf.*, (Sirosh and Miikkulainen, 1994) and (Földiák, 1990)).

[4] All parameter values were found through trial and error.

[5] As far as this implementation is concerned the number of regions and their inter-connectivity is a phylogenetic issue which has been designed here, although, an evolutionary process might alternatively have been used.

4.1.1 Content of Representation

What information is represented by the neural network, the content of the neural code, will be influenced by the learning algorithm and particularly by the learning rules that are used.

From an information theoretic point-of-view any code which preserves the information content of the input data is reasonable and optimal codes are those which are most compact[6]. However, although compact codes may provide efficient communication they require that the receiver contains pre-encoded information about how to decode the message (usually by reconstruction of the original data). Information theory thus considers the faithful transmission of a signal but not its meaning or interpretation. The coding of the data that results may not necessarily be the most relevant, informative, discriminatory or meaningful (Schraudolph and Sejnowski, 1992; Intrator, 1995; Hyvärinen, 1999).

An alternative approach is to generate a code which explicitly represents meaningful aspects of the input data. In the physical world such a code might represent actions, concepts, or objects ('events' in general). Such representations can be learnt since the aspects of the input which constitute an event (for instance, the perceptual properties of an object, *e.g.*, furry, barks, has tail, four legs (Földiák, 1992)) are much more likely to occur together than would be probable if they were not connected. The co-occurrence of input values thus suggests that there may be a causal relationship between them (O'Reilly, 1998) or a common underlying cause for them. Hence, "objects could be defined as conjunctions of highly correlated sets of components that are relatively independent from other such conjunctions" (Földiák, 1990). This assumes that the environment is a conjunction of relatively independent events and to learn to encode such features requires finding independent sources in the input data[7]. This latter approach reduces redundancy by representing individual events with as few active nodes as possible, while compact coding reduces redundancy by representing data using as few nodes (in total) as possible.

Barlow (1972, 1990) has called the co-occurrence of events which happen with greater probability than would be expected by chance "suspicious coincidences", and has suggested that in order to generate representations useful for acting in the world such sets of re-occurring patterns should be represented (Földiák, 1992; Barlow, 1972, 1994; Edelman and Duvdevani-Bar, 1995). To represent a correlated set of inputs it is necessary for a node to learn to respond to those inputs by forming strong synaptic weights to them. In this way multiple inputs related to the same event can be associated together to provide recognition for that event. In addition, by associating different sets of inputs together, which have been generated by the same event perceived under different conditions, an invariant representation

---

[6] One such coding scheme, principal components analysis (PCA), is closely related to unsupervised learning since a linear neuron using a stabilised Hebbian learning rule (section 4.2.1) will learn the first principal component of the input space (Oja, 1982; Schraudolph and Sejnowski, 1992). Further principal components can be found by other nodes if there is a suitable method of competition to cause nodes to represent different components.

[7] An information theoretic approach is independent component analysis (ICA) (Oja, 1995; Hyvärinen, 1999).

can be formed (see section 4.2.2). Similarly, by associating different sets of inputs together, which have been generated by related events, a more abstract classification can be found.

Physiological evidence provides support that neurons in the cortex form representations of meaningful events. The receptive fields of simple cells in the primary visual cortex of mammals are similar to the independent components of natural images (Field, 1994; van Hateren and Ruderman, 1998; Bell and Sejnowski, 1997a,b). In addition, physiological recordings, which find correlations between cell activities and physical stimuli, support this view (see section 3.2.2). In fact, in attempting to record the response properties of single cells, experimental physiology implicitly assumes that individual neurons have functional correlates (Földiák and Young, 1995), and hence the entire enterprise of electrophysiology supports this type of coding. Furthermore, there is little evidence that compact codes are used in the neocortex and there appears to be no reduction in the number of dimensions used to encode sensory data in transforming sensory inputs into cortical representations (van Essen and Deyoe, 1995). Representing independent components is likely to retain or increase the dimensionality of the data but minimises the number of active dimensions representing any input (Hyvärinen, 1999; Fyfe and Baddeley, 1995).

4.1.2 Format of Representation

How information is represented by the neural network, the format of the neural code, will also be determined by the learning algorithm and particularly by the competition between the nodes.

A large number of labels have been introduced into the literature to classify neural codes (*e.g.*, distributed, dense, sparse, local, factorial, coarse, ensemble, topological, population, place, frequency). Multiple classifications can usually be applied to the same encoding and further confusion results from inconsistent usage[8] and by the use of one description to imply other properties (*e.g.*, the term sparse is often used to describe a factorial code even though other forms of coding can also be sparse and there is no necessity for a factorial code to be sparse). Examples of different coding formats are shown in figures 4.3 and 4.4.

Ensemble coding represents input patterns using unique combinations of active nodes. It is the activity pattern of the network as a whole that forms the representation and individual nodes often do not have a meaning in isolation (a node's meaning varies depending on the activity of other nodes). Hence, in ensemble coding it is not possible to ascribe meaning to nodes in isolation. The same arguments as used against compact codes (see above) thus apply, and this format of code does not appear compatible with the known response properties of cortical cells.

While in ensemble coding it is the network as a whole which provides the representation, other

[8] Hence, the definitions given here may not exactly correspond to the usage elsewhere.

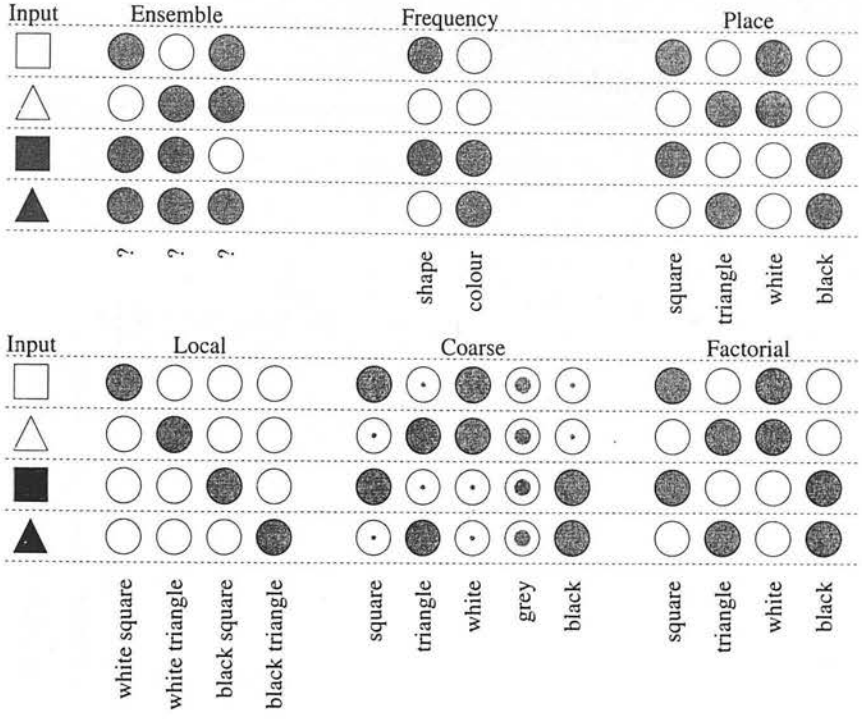


Figure 4.3: Examples of neural coding schemes applied to shapes. Examples of networks encoding four input patterns using different formats are shown. The four input patterns are shown in the left-hand column, these consist of black and white squares and triangles. The response of the nodes in each network to each pattern is shown in the corresponding rows. Each circle represents a node, filled circles represent active nodes. Networks using ensemble, frequency and place coding are shown (top row). Networks using different formats of place coding – local, coarse (soft local), and factorial (population) – are shown (bottom row). (Adapted from Thorpe, 1995)

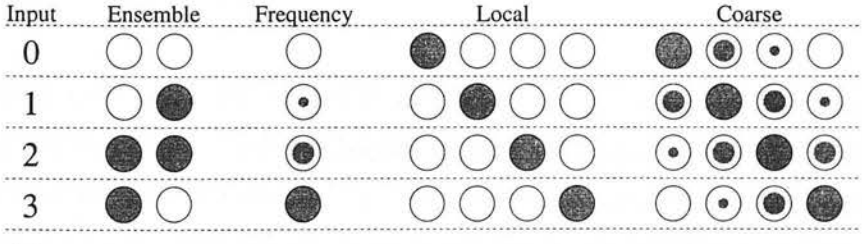


Figure 4.4: Examples of neural coding schemes applied to numbers. Examples of networks encoding four input patterns using different formats are shown. The four input patterns are shown in the left-hand column, these consist of the numbers 0 to 3. The response of the nodes in each network to each pattern is shown in the corresponding rows. Each circle represents a node and the size of the filled circle represents the activation of the node. Networks using ensemble (compact), frequency, local and coarse coding are shown.

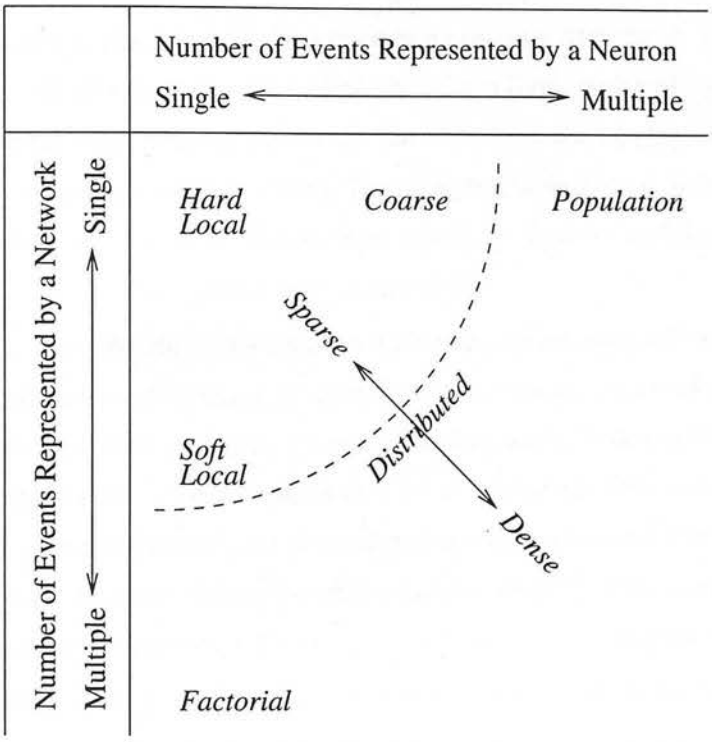


Figure 4.5: Forms of place coding. The figure shows a taxonomy of formats of place codes. The main axes classify coding strategies in terms of the number of different patterns that a single neuron may participate in representing at various times, and the number of patterns that may be represented simultaneously by the network. The sparsity of the code (the proportion of active neurons) increases towards the top-left corner and decreases (or equivalently the density increases) in all directions away from the top-left corner. Codes away from the top-left corner may also be described as distributed.

codes use the individual nodes within the network to provide the representation. For an individual node both its identity and its activation level can be used to encode information. Frequency coding uses an individual node to represent a single variable and uses the activation of the node to indicate the value of that variable (Medler, 1998). Such a scheme is common in subcortical neurons but is not seen in the cortex (except possibly in pyramidal cells in layer V (Miller, 1996); these cells provide output to subcortical structures such as the brain stem and the spinal cord). In contrast, place coding uses the node's identity to represent a particular instance of a variable (Medler, 1998). In place coding it is possible to ascribe meaning to nodes in isolation: nodes are more independent. Hence, the same arguments used in favour of suspicious coincidence detection (see above) apply and this format of code is supported by neurophysiology.

Several different formats of place coding are possible since single or multiple features may be represented at one time using single or multiple active nodes (figure 4.3(bottom row), figure 4.4(right), and figure 4.5). In a distributed code (a population or factorial code) the activity of a group of nodes provides the representation. Depending on the size of this active group, in relation to the total number

of nodes, this may be a dense or sparse code[9]. In the limit a sparse code becomes a local code in which a single node provides the representation as the only active node (through hard, or winner-take-all, competition). With soft competition the winning node only partially inhibits the activity of other nodes, and hence results in other features of the input being partially represented. Coarse coding involves the collective activity of a set of neurons each of which has a single preferred input stimulus but is broadly tuned to respond over a range of inputs such that nodes have overlapping receptive fields (Salinas and Abbott, 1995; Abbott, 1994; Mel, 1990a). The strength of activation of such a node corresponds to the degree of match between its RF and the input pattern, as opposed to frequency coding in which the strength of activation represents the value of the variable.

With a distributed code it is possible that different nodes represent different aspects of a single input pattern (population/coarse coding), or equivalently, the simultaneous presentation of multiple stimuli (factorial/soft local coding). Hence, the same encoding might be described as population or factorial depending on whether it was interpreted as a set of subfeatures representing a single event (a white triangle), or a representation of a set of simultaneous events (white and triangle). The difference is one of interpretation. In the former case (population/coarse coding) the input space is considered continuous, in the later case the (factorial/soft local coding) the input space is considered discrete.

In contrast to a distributed code, in which multiple nodes may be active to represent multiple features of the input, a local code represents each possible input using a distinct node and is hence restricted to representing mutually exclusive events. The simultaneous presentation of multiple input features can only be represented by a single node, requiring distinct 'grandmother cells' to represent each combination of inputs.

Local codes effectively generate a look-up table, in which there is no interference between representations but also no generalisation and limited capacity. Dense codes do provide generalisation and have higher capacities but interference between representations may cause problems with learning. Sparse codes combine the best feature of both local and dense codes (Földiák and Young, 1995; Felman, 1990). These computational advantages make sparse codes the most likely coding strategy of the cortex (Földiák and Young, 1995; Olshausen and Field, 1996b,a; Field, 1994; Barlow, 1995).

In addition to the physiological evidence, place coding is also to be preferred as it has computational advantages over both frequency and ensemble coding. In a place code, since the dimensionality of the input is greater and the node responses are more independent, distinct input patterns can be represented by orthogonal codes. The number of ways of linearly partitioning the data is thus increased. In this way implicit properties of the data are made more explicit *i.e.*, relational problems are transformed into

---

[9] For neurons with binary activation states the sparseness can be measured simply as the ratio of active neurons to inactive ones at each time step. For neurons with continuous output functions sparseness may be measured using the kurtosis ('peakedness') of the activity distribution of cells (Field, 1994; Harpur and Prager, 1996; Fyfe and Baddeley, 1995).

The Repetition of Values																					
Frequency Coded				Ensemble Coded						Place Coded											
0	0	→	1	0	0	0	0	→	1	0	0	0	1	0	0	0	1	→	1
2	3	→	0	1	1	1	0	→	0	0	1	0	0	1	0	0	0	→	0
1	1	→	1	0	1	0	1	→	1	0	0	1	0	0	0	1	0	→	1
1	2	→	0	0	1	1	1	→	0	0	0	1	0	0	1	0	0	→	0

(a)

The Difference Between Values																					
Frequency Coded				Ensemble Coded						Place Coded											
0	1	→	1	0	0	0	1	→	1	0	0	0	1	0	0	1	0	→	1
0	2	→	0	0	0	1	1	→	0	0	0	0	1	0	1	0	0	→	0
3	2	→	1	1	0	1	1	→	1	1	0	0	0	0	1	0	0	→	1
1	1	→	0	0	1	0	1	→	0	0	0	1	0	0	0	1	0	→	0

(b)

Table 4.1: Examples of recoding relational problems. Small samples of sets of training data are shown for two problems. (a) The output value is 1 when the two inputs have the same value. (b) The output is 1 when the absolute difference in value between the two inputs is exactly equal to 1. The problems are shown using different coding strategies for the input data. Changing the coding strategy can convert a relational problem into a statistical one. In both examples place coding is the only scheme in which the input can be used to directly justify the output (*e.g.*, in example (a) if both the 4th and 8th inputs, or the 3rd and 7th inputs, are 1 then the output should be 1; in example (b) if both the 4th and 7th inputs, or the 1st and 6th inputs, are 1 then the output should be 1).

statistical problems (see table 2.1 in section 2.3.3.1 for a definition of statistical and relational problems, and table 4.1 for an example of recoding). Learning more complicated, relational, properties is enabled by using recodings which convert the problem into a simpler, statistical, learning problem over the recoded data (Clark and Thornton, 1997; Thornton, 1997, 1996a, 1995, 1996b). Place coding can thus improve learning since correlations between patterns can be found in the statistical effects between data values. Regularities encoded in the relationship between two or more input variables should thus be recoded so that the relationship is expressed as a single variable *i.e.*, input variables should be made independent. In the limit, when purely local coding is used, every input pattern must be represented by a unique node, hence it is a trivial problem to associate this unique node with any desired response. However, better generalisation may be possible if the encoding is more distributed than a purely local code. Hence, the most appropriate level of abstraction is likely to be found by a process of recoding data (finding more abstract, more locally coded, representations) until the most appropriate level of abstraction has been found. Similarly, Field (1994) proposed that the goal of sensory coding is to produce sparser and sparser representation as the processing proceeds to higher cortical areas and Barlow (1960) proposed that each level of sensory processing reduces redundancy. This process should be able to be achieved by applying the same mechanisms, of recoding and finding associations, to the ever more abstract representations at each level (Clark and Thornton, 1997; Thornton, 1997, 1996a,b, 1995). The representations at each level of abstraction constrain the search space for subsequent abstractions and hence enable deeply buried regularities to be uncovered (Clark and Thornton, 1997).

The above argument suggests that knowledge can be gained by recoding information that is already stored in internal representations (Karmiloff-Smith, 1994). A similar argument was made in section 2.2.2 for the need to find the correct level of abstraction so as to make the task tractable (too low-level and too many connections need to be used, too high-level and too many nodes need to be used). Hence, although raw image data provides a distributed code capable of representing every possible image, more abstract representations of objects are required but not such abstract representations that every image is coded separately. In addition it was argued that the correct level of abstraction was required to convert sensory data that contains implicitly encoded information into explicit representations appropriate to provide direct justification for an output. Such internal recodings can be considered as (perfectly aligned) 'virtual sensors' and used to signal external events (sections 2.1.1.2 and 2.2.2, Thornton, 1996a, 1997).

Hence, events which occur sufficiently frequently, or have sufficient behavioural significance, ought to be represented using a more abstract (*i.e.*, more local) code (Page, 2000; Wallis and Bülthoff, 1999). Other events may be represented using a distributed encoding. The process of automation (section 3.1.1.3) might be due to forming a more local representation to supplement or replace an initially distributed one (section 3.2.2). The accompanying improvement in task performance does not generalise (Karni et al., 1998) which is consistent with the formation of a local code. As representations become more abstract (more local) there is less interference between patterns and faster learning rates can be used (Plunkett et al., 1997). Hence, fast (episodic) learning might rely on more local coding (Plunkett et al., 1997) but may also rely on a separate memory storage mechanisms involving the hippocampus (McClelland et al., 1995; Janowsky, 1993).

There are thus good biological and computational reasons to favour the use of place coding. However, each of the formats of place coding needs to be possible; both distributed and local (Page, 2000). In addition, topological maps are a ubiquitous feature of cortical representations (section 3.2.2) and ought to be modelled. A topological representation modifies the spatial organisation of receptive fields rather than their properties, and hence might be used in conjunction with any of the forms of coding described above.

Since there is evidence to suggest that neurons, from all areas of the neocortex, apply the same learning mechanisms (section 3.2.3), it would seem to be important to have a single artificial neural network model which can generate all these forms of representation. The model thus needs to be able to develop representations which are place coded and able to represent both single and multiple events in both the discrete and continuous domains, using either topologically or non-topologically organised nodes.

Within the diversity of unsupervised learning algorithms which have been proposed all these forms of representation can be learnt, however, no single algorithm learns them all. Hence, none of these

algorithms have the required properties. In addition, the application of algorithms which learn a specific form of representation requires that the appropriate form of encoding for a particular task be known beforehand in order to select an appropriate algorithm. Section 4.2 describes a single learning algorithm which can generate local, distributed and topological representations, as appropriate to the structure of the input data received. The format of neural coding depends predominantly on the competition between the nodes and the algorithm described here relies greatly on a novel form of lateral inhibition (section 4.3.2). The learning rule presented in section 4.2.1 is designed to represent frequently occurring groups of inputs, and hence to learn suspicious coincidences. The neurons in the network will thus form feature detectors.

4.1.3 Usage of Representation

The previous two subsections have discussed requirements for a neural network to be able to generate useful encodings in terms of what is represented and how it is represented. This subsection shows how the proposed architecture will make use of networks which can learn such representations.

Up to now we have considered an abstract notion of the input to a network. An obvious, concrete, source for such inputs is from sensory systems. However, the nervous system "does not distinguish between perturbations from outside and internal perturbations, since nervous signals are unspecified with regard to their origin" (Riegler, 1994b). Hence, the outputs of primary sensory neurons and any other type of neuron must be treated in the same way. Sensory data provide patterns of activations which get represented by neural activations. These neural activations are themselves patterns of activations which can in turn be represented by further neural activations. The same processes which develop useful codings for sensory data can thus also be applied to developing useful codings of neural data, and so on hierarchically, provided that (a) the output of one network provides (part of) the input to other networks, and (b) that each network can work with the same coding format for both inputs and outputs. The neural network that is presented here generates local, distributed and topological representations and can process inputs encoded in any of these formats. Hence, by connecting such neural networks together it is possible to form a modular assembly in which each network is initially identical but learns, via the application of identical learning rules, to represent the data it receives.

Stylised examples of the kind of assemblies envisaged are shown in figure 4.6. Such assemblies can be interpreted as models of pathways through cortical regions in which areas are pre-encapsulated. However, the model is purely indicative rather than representative of particular cortical regions and the connectivity between them. Note that arrows between regions represent full connectivity between the outputs of all nodes in the source region and the inputs to all nodes in the target region. Hence, networks (like cortical regions) do not constitute closed processing modules with hidden, local, representations (Mumford, 1994).

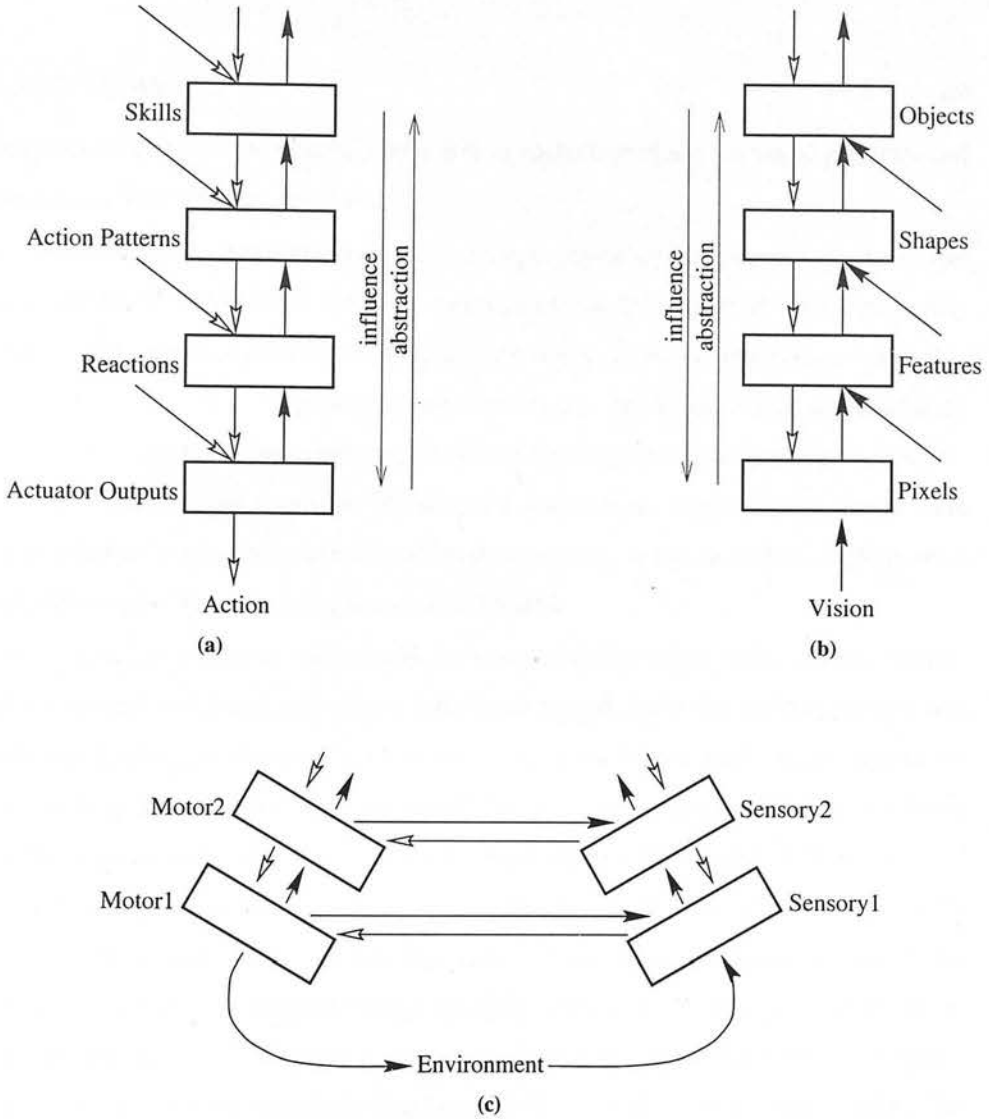


Figure 4.6: Schematics of examples of assemblies of neural networks. (a) An example of a possible motor hierarchy. (b) An example of a possible sensory hierarchy. (c) An example of a possible sensory-motor assembly.

The same architecture is used for processing abstract representations as for simple ones. Neural events within the assembly are processed in the same way as neural events produced in response to sense data. In moving from region to region the same processing is applied to information travelling along a pathway. Such repeated processing could discover more abstract representations of the input data (*e.g.*, in the visual system, a region receiving physical stimuli could abstract primitive features, and a region receiving the primitive features could learn object representations (Földiák, 1992)) and integrate information from different domains[10] and form larger receptive fields. Cortical cells do become selective

[10] Data is not represented as coming from a particular sensory domain (it is amodal). Hence, there is no difference between sensory integration between domains and sensory abstraction within the same domain (contra Ghahramani et al., 1997): forming a

to more complex features and have increasingly large RFs at each stage along the visual pathway, and do integrate information from different domains.

No further processing can be done on the output of a single region which generates a purely local code, as only one event is ever represented. However, correlations with the outputs of other regions may still be found. Further abstractions within the output of a single region, as well as correlations with other regions can be found for a network which generates a distributed code. Topological maps are found in all regions of the cortex. In order for a subsequent region to form topologically organised representations, similar events in the preceding region must be encoded in a similar way. Coarse coding would seem to be necessary in order to transfer topologically ordered information between cortical regions since similar events are represented by overlapping sets of active nodes.

Primary sensory and motor regions interact with the external environment. Other regions interact with an internal environment of neural activations. It has been argued above that learning to represent the internal environment will generate more abstract representations. Motor output in reaction to the internal environment may thus generate more complex behaviour than simple sensory-motor reflexes in reaction to direct sensory cues; complex acts require more abstract representations (Morasso and Sanguineti, 1994). Complex behaviour would thus be explained as reactive behaviour within a richer environment provided by the internal representations (the knowledge, memory and experience) of the agent. Complex behaviour can thus produced through the same mechanisms used to generate simple reactive behaviour. Simple reactive behaviour is in response to external triggers (from both exteroception and proprioception), while complex behaviour is in response to internal triggers (section 3.1.1). Internal triggers are provided by more abstract representations of external stimuli, while external triggers are themselves represented by (less abstract) internal neural events. Hence there is no real distinction between internal and external stimuli, but rather a spectrum from less to more abstract behavioural cues. The model allows adaption and reaction to both external stimuli and internal stimuli. It is thus situated at all levels within the environment.

At the lowest level each motor neuron must have some predefined action. Higher levels ought to group these primitive actions into higher-level behaviours. Higher levels of the hierarchy would then need to be able to influence lower levels so that events within the assembly can influence actions. The choice of these primitive actions will affect subsequent learning: higher-level innate capacities would reduce the search space for more complex behaviours and so speed-up learning, but would reduce the variety of possible behaviours (*cf.*, the bias-variance problem; section 2.3.3.1).

The representations formed in any one region do not need to be decoded, from the point-of-view of the neural assembly itself (Morasso et al., 1998), since they can have direct effects on the activities of other nodes through synaptic connections. In effect, the decoding is provided by the connections

---

domain independent representation can be done in the same way that an invariant representation in one modality is formed.

between representations which are learnt at the same time as the representations themselves. The semantics of one representation is the effect it has on other representations. Because there are no ascribed semantics the system is grounded. For an external observer the neural code can be decoded by measuring the preferred input of each node. The preferred input may be found by recording the response of the node to stimuli[11], or (for artificial neurons) by examining the synaptic strengths[12]. Such measurements can be used to interpret the activation of individual nodes. The simplest means of decoding the activation of multiple nodes, which all respond to the same input (*i.e.*, a coarse or population code), is the vector method (Salinas and Abbott, 1995) in which the normalised sum of the activations weighted by the preferred input of each node is calculated: $stimulus = \frac{\sum_{i=1}^{n} y_i Pref_i}{\sum_{i=1}^{n} y_i}$, where y_i is the output activation of node i, and $Pref_i$ is the preferred stimulus (the centre of the receptive field) of node i.

4.2 Learning Rules

4.2.1 Qualitative Analysis

Learning rules define how the synaptic weights are changed in response to each pattern presented to the network. To be biologically plausible such rules should refer only to information available locally at a synapse (although information might arrive at a synapse through a variety of mechanisms from sources other than the pre- and post- synaptic cell). Several rules for use in unsupervised learning have been suggested which calculate weight changes as a simple, linear, function of pre- and post-synaptic activations. These rules are summarised in table 4.2. All of these rules increase the synaptic weight if both the pre-synaptic and post-synaptic activations are high at the same time, and hence provide the type of learning proposed by Hebb (Hebb, 1949; Brown and Chattarji, 1995). All these rules can thus be considered as variations of Hebbian learning.

The Hebbian learning rule states that synaptic weights should be strengthened when there is correlation between pre- and post-synaptic activity. This is in accordance with conditions for the long-term potentiation (LTP) in biological neurons (Eglen, 1997; Brown and Chattarji, 1995). The simplest version of this rule increases the synaptic strength in proportion to the product of the pre-synaptic and post-synaptic activities:

$$\Delta q_{ij} = \beta x_j y_i.$$

The Hebb rule is unstable since there is a positive feedback effect causing the increase in q_{ij} to become

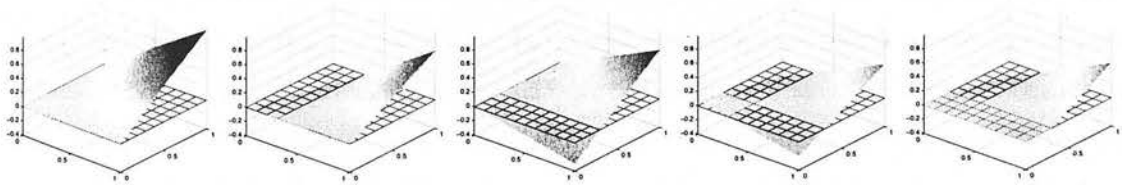
---

[11] Bayes' theorem could then be used to find the probability that a particular stimulus occurred given the neural activities (Földiák, 1993; Oram et al., 1998).

[12] To find the preferred input for a node in a region not receiving input directly from the sensory array this might require recursively decoding synaptic weights at each level until the input sources are reached. This will only provide an approximation to the true RF for nodes with nonlinearities in their activation functions (Wallis, 1994).

Synaptic Activity		Hebb	Instar	Outstar	Covariance
x (pre-)	y (post-)	$\Delta q \propto xy$	$\Delta q \propto (x-q)y$	$\Delta q \propto x(y-q)$	$\Delta q \propto (x-\tilde{x})(y-\tilde{y})$
high	high	+ (large)	+ (large)	+ (large)	+
low	high	+ (small)	− (large)	+ (small)	−
high	low	+ (small)	+ (small)	− (large)	−
low	low	+ (tiny)	− (small)	− (small)	+

Table 4.2: Qualitative behaviour of simple pseudo-Hebbian learning rules. This table provides a qualitative comparison of the weight changes that would occur for each learning rule under each condition of the pre- and post-synaptic activity. This assumes that intermediate values for the thresholds are used. The instar and outstar rules use the current value of the synaptic weight as the threshold. The covariance rule may be used with arbitrary values as the thresholds (Eglen, 1997), or using the long-term averages of the activity values (Sejnowski, 1977).



(a) (b) (c) (d) (e)

Figure 4.7: Comparison of learning rules. Figures show the change in synaptic weight (on the vertical axis) caused by each rule, using intermediate values for thresholds, for varying values of pre-synaptic activation (on the ↘ axis) and post-synaptic activation (on the ↗ axis). (a) Hebbian learning rule, (b) instar learning rule, (c) outstar learning rule, (d) covariance learning rule, (e) proposed learning rule.

larger as q_{ij} increases (since increasing q_{ij} will increase y_i). In addition, weights can increase without bound. Simply placing a maximum limit on q will result in the eventual saturation of all weights, and the loss of receptive field selectivity, due to the repeated application of this weight increasing rule. Normalisation of the synaptic weights is effective at stabilising this rule. There are several possible variations in the manner that normalisation is applied. Normalisation may be applied to the total synaptic weight impinging on a node (post-synaptic normalisation; von der Malsburg, 1973) or to the total synaptic weight from an input source (pre-synaptic normalisation; Eglen, 1997). Normalisation may be applied divisively or subtractively (in divisive normalisation each weight is divided by an equal value, in subtractive normalisation an equal value is subtracted from each weight) with differing results (Goodhill and Barrow, 1994; Miller and MacKay, 1994). Normalisation can be implemented explicitly (von der Malsburg, 1973) or by use of a decay term which causes the weights to approach normalised values slowly over time (Oja, 1982; Miller and MacKay, 1994). It is most common to use divisive normalisation of the input weights, *e.g*:

$$q_{ij} = \frac{q_{ij}+\beta x_j y_i}{\sum (q_{ij}+\beta x_j y_i)} \quad \text{explicitly, or equivalently;}^{13} \quad \Delta q_{ij} = \beta \left(x_j y_i - q_{ij} y_i^2 \right).$$

Normalisation has other useful properties in addition to preventing unbounded weight increases. Nor-

[13] This rule actually causes normalisation of the sum of the squares of the weights, rather than the sum of the weights.

malisation helps to prevent any single node coming to represent a disproportionately large region of the input space. Normalisation also has the effect of providing competition between nodes, in addition to that provided by the lateral inhibition, since it strengthens connections to one part of the input space while weakening connections to another (Swindale, 1996). However, normalisation is not used in the algorithm described here. Instead, 'habituation' (section 4.3.1) is used to prevent any nodes from representing a disproportionately large region of the input space, and activity-dependent decreases in synaptic strength are used to refine the receptive fields (this section, below). Not using normalisation enables all synaptic weights to be initialised with zero strength.

The Hebb rule does not consider the consequences of uncorrelated activity, nor conditions under which synaptic weights could decrease (Brown and Chattarji, 1995). The instar rule modifies Hebbian learning to allow activity-dependent reduction in synaptic strength if the pre-synaptic activity is below a threshold (Marshall, 1995a; Kalarickal, 1998). This is in accordance with conditions for the heterosynaptic long-term depression (LTD) (Eglen, 1997; Frégnac, 1995). The outstar rule provides for activity-dependent reduction in synaptic strength when the post-synaptic activity is under a threshold (Marshall, 1995a; Kalarickal, 1998), in accordance with homosynaptic long-term depression (Eglen, 1997; Frégnac, 1995). The covariance rule (Sejnowski, 1977) differs from the instar and outstar rules in using the long-term average, or trace, of the pre- and post-synaptic activations as the thresholds, rather than the current value of the synaptic weight. The covariance rule uses both forms of weight reduction, however, it also generates weight increase when both pre- and post-synaptic activity is low simultaneously. Such an increase in synaptic strength, when neither input nor output is active, would cause a network to increase all weights when no input was supplied (Grzywacz and Burgi, 1998), and for a network using sparse coding most weights will be increased at each iteration. Such weight increases are not specific to any correlation in activity and are thus undesirable. This is easily resolved by preventing weight increase when both pre- and post-synaptic components are negative (*e.g.*, Montague et al., 1991; Barlow, 1990; Grzywacz and Burgi, 1998). Another potential problem with all three of these rules, if biological plausibility is required, is that synaptic weights can change sign. This also can easily be resolved by placing a limit on q at zero (*e.g.*, Grzywacz and Burgi, 1998).

The learning rule that has been employed throughout this thesis is of the form:

$$\Delta q_{ij} = \beta \, (x_j - \tilde{x})(y_i - \tilde{y}_i)^+ \tag{4.3}$$

where $(v)^+$ is the positive half-rectified value of v, and q_{ij} is constrained not to change sign (*i.e.*, $q_{ij} = (q_{ij})^+$). As with the covariance rule the long-term average, or trace, of the pre- and post-synaptic activations are used as the thresholds:

$$\tilde{x} = \tau_x \bar{x} + (1 - \tau_x)\tilde{x} \tag{4.4}$$

$$\tilde{y}_i = \tau_y y_i + (1 - \tau_y)\, \tilde{y}_i \tag{4.5}$$

Synaptic Activity		Learning Rule
x (pre-)	y (post-)	$\Delta q \propto (x - \tilde{x})(y - \tilde{y}_i)^+$
high	high	+
low	high	−
high	low	0
low	low	0

Table 4.3: Qualitative behaviour of the preferred learning rule. This table provides a qualitative indication of the weight changes that would occur for the learning rule used in the algorithm described here, under each condition of the pre- and post-synaptic activity. This assumes intermediate values for the thresholds are used.

where $\bar{x} = \texttt{mean}_j\{x_j\}$ is the mean value of the input activities at the current iteration. The behaviour of this learning rule is given in table 4.3 which may be compared to that of the other learning rules shown in table 4.2 and using figure 4.7. This rule is most similar to the covariance rule, except the post-synaptic term is only allowed to be greater than or equal to zero. Hence, post-synaptic activation exceeding a threshold is required before synaptic modification is enabled. Such depolarisation will usually require the synchronous discharge of several converging afferents and thus strengthening occurs of 'cooperating' coincident synapses (Bear, 1985) together with weakening of synapses to inactive input sources. Sub-threshold post-synaptic activity does not result in any changes in synaptic strength. This restriction allows nodes to represent stimuli which share input features. To illustrate why this is necessary consider a simple network consisting of two nodes receiving three inputs (a, b and c), and input data consisting of the patterns 'ab' and 'bc' (figure 5.15). If the post-synaptic term in the learning rule was allowed to go negative then the node representing pattern 'ab' would decrease the strength of the synapse to input b whenever pattern 'bc' was presented. Similarly, the node representing pattern 'bc' would decrease the strength of the synapse to input b whenever pattern 'ab' was presented. The result would be that input b would not be strongly connected to either node. If 'b' was a subfeature of many patterns then the weight decreases would exceed the increases and no connections from input 'b' would be learnt. Similarly, if 'a' was also a subfeature of several patterns then connections to 'a' would also not form. No representation of pattern 'ab' would then be possible. It is thus essential to prevent the post-synaptic term of the learning rule from being negative in order to represent any discrete space where patterns have common subfeatures or for mapping any continuous space of two or more dimensions. Receptive fields are differentiated by the reduction in synaptic strength provided when the pre-synaptic activity is sub-threshold. This forces nodes to represent one set of co-occurring features. When using a non-linear node (section 4.4.2), which is designed to represent multiple sets of co-occurring features, the pre-synaptic term is also restricted to be positive or zero.

Although the equation given above provides a qualitative description of the learning rule that is

used the actual learning rule employed is more complex:

$$\Delta q_{ij} = \beta \frac{m\left(\frac{x_j}{\tilde{x}} - 1\right)}{\sum_{k=1}^{m} \left|\frac{x_k}{\tilde{x}} - 1\right|} \frac{n\left(\frac{y_i}{\tilde{y}_i} - 1\right)^+}{\sum_{k=1}^{n} \left(\frac{y_k}{\tilde{y}_k} - 1\right)^+}. \tag{4.6}$$

The pre- and post-synaptic terms have been rearranged so that weight changes are proportional to the ratio of the current activity to previous activity, rather than the difference. This modification favours higher weight increases to nodes which have lower average activation values, *i.e.*, lower $\tilde{y}_i$, (caused by either being active with low strength or with low frequency). Hence, this modification more forcibly encourages all nodes to be equally active. It has no effect on the pre-synaptic term since all synapses use the same threshold value, $\tilde{x}$, and hence:

$$\frac{m\left(\frac{x_j}{\tilde{x}} - 1\right)}{\sum_{k=1}^{m} \left|\frac{x_k}{\tilde{x}} - 1\right|} \equiv \frac{m\left(x_j - \tilde{x}\right)}{\sum_{k=1}^{m} \left|x_j - \tilde{x}\right|}.$$

The pre-synaptic term is simply rearranged for symmetry with the post-synaptic term. In addition the pre- and post-synaptic terms are normalised so that the total change in synaptic weight that can occur at each iteration is normalised across the network and for individual dendrites. This gives equal importance to each training pattern[14], and also allows nodes with small receptive fields to compete with nodes having large receptive fields, since the total synaptic weight change in both cases will be the same. In addition, scaling each term by the number of items in the normalisation means that synaptic weight changes are independent of the number of inputs or the number of nodes. The algorithm is thus stable to changes in these values, and hence, is robust for use in different regions in an assembly and for application to different tasks. Furthermore, the learning rate does not depend on the absolute size of the activations. This enables all synaptic weights to start at zero strength without the initial learning rate being made slow due to the small initial strength of the post-synaptic activation (learning rates do not depend on the absolute size of the post-synaptic activation)[15]. Similarly, since one region may receive its inputs from another region (which will also be gradually increasing its activation) the learning rate should not depend on the absolute size of the pre-synaptic activation either. Normalisation of the learning rule provides limits on the synaptic weight changes that can occur in any one learning step, however, it does not produce weight normalisation since weight changes for each node will differ. Such normalisation of learning could be interpreted as a limit on the availability of a resource in the region, and is at least as justifiable as the explicit normalisation of weights used to stablise Hebbian learning.

Unlike many other neural network models all synaptic weights start at zero strength. Most other neural network models give random initial values to the afferent synapses. Random weight initialisation

[14] If no node is sufficiently active to have a positive learning rate, when there is an input, then the threshold is reduced (from $\tilde{y}_i$ to $\bar{y}$) to force some learning to occur. This attempts to ensure that all inputs become represented, however, it is not very effective and has very little affect in practice.

[15] Division by zero, when $\tilde{x} = 0$ or $\tilde{y}_i = 0$ at the start, is avoided by defining $\left(\frac{v_j}{0} - 1\right) = 1$ (an arbitrary finite value).

enables weight normalisation to be used, and also causes y to be greater than zero at the start so that learning can occur: for all the learning rules in table 4.2, if $\sum w = 0$ then $y = 0$ and hence $\Delta q = 0$ and all weights remain at zero. For a real cortical region it would be inefficient to form random connections of arbitrary weight prior to experience invoked neural activity, only for these connections to be substantially modified by the subsequent input activity. The output of such a region would also be incorrect until considerable reorganisation had taken place. There seems to be neither biological nor computational justification for using random initial weights. In contrast, the growth of synapses from an initial strength of zero makes the output representation inaccurate, rather than wrong, during learning and enables representations to be learnt more quickly using positive training examples only (as occurs in language learning). In addition, synaptogenesis could be modelled although in the current model neither dendritic growth nor synaptic elimination is considered. However, the processes underlying synaptic sprouting and pruning are likely to be activity dependent and hence similar to the processes responsible for synaptic strengthening and weakening (Miller, 1998; Quinlan, 1998; Elman et al., 1996), and may in fact be part of the same process since stabilising or retracting connections will require strengthening and weakening of weights (Miller, 1998). Hence, there may be no need to model structural modifications separately from synaptic ones, and a model performing synaptic weight modifications within a network with fixed connectivity, as described here, may be sufficient.

In this model a synaptic strength of zero has a slightly different meaning than in most other neural networks: it represents a potential connection rather than no connection. In order to represent no connection it would be necessary to not instantiate a variable for that weight. In this way a bias in connectivity, such as an arbor function, could still be achieved. However, the current implementation does not allow an arbor function to be used due to the manner in which lateral inhibition is implemented; in order to efficiently implement the simplification provided by equation 4.13 the algorithm assumes that all laterally connected nodes receive the same input connections in the same order. This is a technical limitation of the particular implementation that has been used rather than the algorithm itself. The corollary of this is that there needs to be full lateral connectivity between all nodes in a region (to provide competition, section 4.3) since all nodes are connected to the same inputs and can potentially represent the same input data.

To prevent synaptic strengths increasing indefinitely it would be possible to allow weights to saturate at a maximum level, since the negative weight changes will ensure that only certain weights reach saturation. However, such a method would mean that a node representing a greater numbers of inputs (having a larger RF) could generate a higher activation value than nodes connected to fewer inputs. Instead, the method used rescales the weights to ensure that the maximum activation achieved by any node is the same (*i.e.*, $\hat{y}_i \leq 1$):

$$q_{ij} = \frac{q_{ij}}{\max\left(1, y_i^{out}\right)} \tag{4.7}$$

This method is also not ideal when nodes represent overlapping patterns (see section 5.2). However, it should be noted that this mechanism has no impact on most results since the low value of the learning rate means that few nodes achieve an activation approaching a value of one except after an excessively high number of iterations.

4.2.2 Learning Invariances

So far we have only considered learning to recognise sets of features that occur simultaneously. Such learning relies on statistical regularities within a co-occurring set of inputs which are assumed to be caused by events in the world. However, since the world generally changes slowly, there is also likely to be statistical regularities between consecutive sets of inputs as well (*i.e.*, temporal correlations as well as spatial ones). Temporally extended input sequences provide information about how different stimuli may result from the same event. Consider visual input due to an object: "in the real world we see objects for protracted if variable periods whilst they undergo a number of natural transformations ... The consistent stream of images serves as a cue that all of these images belong to the same object and that it would be expedient to associate them together" (Wallis and Baddeley, 1997). Conversely, it is unlikely that separate events will consistently occur in quick succession if they are unconnected (Wallis, 1994). While static data can be used to classify inputs which are similar, and hence to recognise objects from a single viewpoint, temporal sequences of data can be used to classify input patterns which are dissimilar but which have the same underlying cause. This enables abstract representations to be learnt, which are invariant to viewpoint. Hence, it has been proposed that temporal continuity in the appearance of objects under-pins the learning of invariance (Templeman and Loew, 1989; Földiák, 1991, 1992; O'Reilly and McClelland, 1992; Becker, 1993, 1997, 1999; Wallis, 1994; Wallis and Baddeley, 1997; Wallis and Bülthoff, 1999; Marshall, 1995b).

A simple modification to a pseudo-Hebb learning rule can provide a mechanism which exploits this temporal relationship between stimuli to group those inputs representing the same object (Földiák, 1992). The modification requires that a short-term trace of activity be used instead of the value for instantaneous activity. This enables previous activity to influence learning of subsequent inputs and hence transforms temporal regularities into spatial ones (Wallis, 1994). There are two possibilities, either a trace of the pre-synaptic activity (Földiák, 1992) or of the post-synaptic activity (Földiák, 1991, 1992; Wallis et al., 1993; Wallis and Rolls, 1997; Wallis, 1994, 1996, 1998; O'Reilly and Johnson, 1994; Ebdon, 1996) can be used. A post-synaptic trace will encourage the same node to remain active and hence to associate this output with subsequent inputs. A pre-synaptic trace will cause inputs to remain active for a short time encouraging those inputs to be associated with subsequent outputs. There is biological evidence for various mechanisms that could give rise to either type of trace (Wallis and Rolls, 1997; Wallis, 1998; Földiák, 1992; Wallis and Rolls, 1997). Even for learning spatial coincidences

it must be the case that synaptic modification relies on activity during a time window, rather than on instantaneous values, due to different latencies for convergent pathways in the cortex, and the discrete nature of action-potentials (Ebdon, 1996).

A trace of pre-synaptic activity has been used since it has several advantages over a post-synaptic trace for the particular algorithm described here.

- Since the pre-synaptic term of the learning rule can be negative it is important to use a pre-synaptic trace to generate positive weight changes to inputs which are correlated across time. A post-synaptic trace would only serve to increase negative weight changes to previously active inputs.

- Since all post-synaptic activities are initially similar (due to the synaptic weights all starting at zero) a post-synaptic trace would have little effect initially. Pre-synaptic activity coming from an external input, or from a lower-level region in a more advanced stage of learning, is likely to have much more variability and thus a pre-synaptic trace can be more effective.

- A post-synaptic trace will cause a subset of nodes to increase their synaptic weights over a short period of time. This increase in synaptic weights may result in these nodes being more activated by subsequent inputs. Without weight normalisation there may be a positive feedback effect which causes a subset of nodes to keep winning for all inputs. Other implementations (*e.g.*, O'Reilly and McClelland, 1992) which have used a post-synaptic trace have provided null inputs in between sequences of different objects in order to reset the post-synaptic trace. This limits the positive feedback effect, as well as preventing associations between distinct events, but it is biologically implausible and is not used here.

The trace is implemented by modifying the current input activation at each synapse by a function of its previous activity:

$$x_j = \lambda_x x_j^{in} + (1 - \lambda_x) x_j. \tag{4.8}$$

This is described as sensitising the input activation to previous activations, and λ_x is the sensitisation factor. To provide sensitisation a value of $\lambda_x = 0.5$ is used. In many experiments there are no temporal correlations in the input data, in which case no sensitisation is used and $\lambda_x = 1.0$. This rule is of the same form as that used to calculate the pre- and post-synaptic threshold values (*i.e.*, $\tilde{x}$ and $\tilde{y}_i$, see equations 4.4 and 4.5). However, those values are long-term traces of the activity and are generated using a much smaller time constant (*i.e.*, $\tau_x = \tau_y = 0.001$). Another difference is that the long-term trace of the input activity is a single value for all synapses whereas the sensitised values of the input activations are individual to each synapse.

There need be no similarity between the stimuli that are associated together by the trace rule, and in many cases there will be no overlap between each pattern and hence no similarity at all. Thus, the

similarity of the inputs is irrelevant and this rule could be used to associate arbitrary stimuli occurring sequentially or to associate stimuli resulting from a single object undergoing arbitrary transformations. The experiments in subsequent chapters only consider translation, but this rule is not limited to only learning translation invariance.

The justification for the trace rule is that underlying physical events change slowly; this is true across space as well as time, so that nearby sensors will tend to record similar values (Eglen et al., 1997; Stone and Bray, 1995; Kuipers et al., 1993). Whereas local lateral excitation provides a constraint for spatial smoothness, the trace rule provides a constraint on temporal smoothness. Temporal smoothness is at odds with habituation, while spatial smoothness is at odds with lateral inhibition, since both habituation and competition try to maximise the variance in the responses of individual nodes[16]. It is possible to combine these two requirements in a single objective function which attempts to minimise short-term variance and maximise long-term variance (Stone and Bray, 1995) and to minimise short-range variance and maximise long-range variance (Eglen et al., 1997). Other implementations have also relied on constraints in the objective function (Becker, 1997, 1999). A separate class of methods uses the structure of the neural network itself to provide translation invariance (*e.g.*, Fukushima, 1980, section 6.1.2.3), and hence translation invariance is built in to the neural architecture rather than being learnt.

4.2.3 Timing Considerations

When using an assembly of interacting neural networks timing considerations become important in order to recognise corresponding events in each of the regions. This is particularly important for correct learning, since the learning algorithm attempts to form representations of correlated events. The architecture has been implemented so that regions are updated sequentially. It has been found necessary to use two types of connections between regions: those whose synapses are modified after calculating the new output of the region (*i.e.*, $\Delta q_{ij} = \beta f_3(x_j(t), y_i(t))$); and those that are modified prior to updating the output (*i.e.*, $\Delta q_{ij} = \beta f_3(x_j(t), y_i(t-1))$)[17]. These connection types vary only in the relative timing of the pre- and post-synaptic events which they associate together. These two variants are found to be sufficient to keep learning synchronised in simple hierarchies: allowing the regions to be run sequentially while ensuring that the correct associations are learnt. They will be referred to as afferent and efferent connections, or synapses. These terms are used since in general an afferent pathway (one transmitting information towards more central regions) will use the first variation, while an efferent pathway

---

[16] Competition between nodes is likely to prevent a single node from representing all transformations of a single object. Multiple nodes might thus be needed which are invariant to different transformations of the same object. Such representations are observed in the cortex (Wallis, 1994; Perrett, 1996).

[17] Strictly speaking only the former is Hebbian since the original definition (Hebb, 1949) states that synaptic efficiency is increased when the pre-synaptic cell takes part in activating the post-synaptic cell.

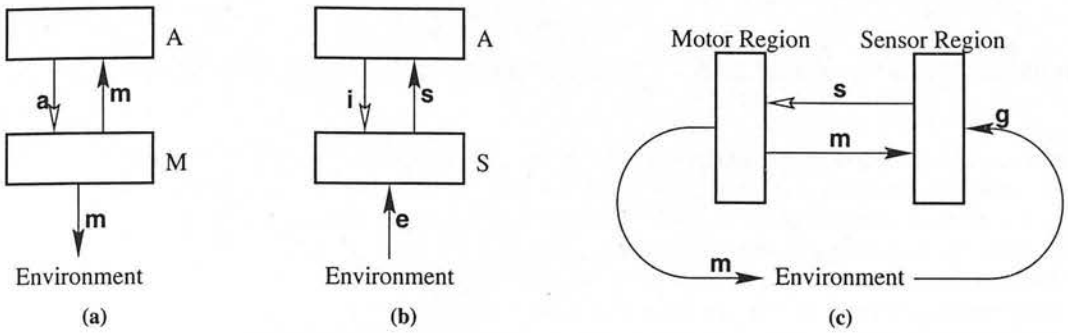


Figure 4.8: Simple regional assemblies. (a) The simplest assembly for testing motor abstraction. (b) The simplest assembly for testing sensor abstraction. (c) The simplest assembly for testing sensory-motor coordination.

(one transmitting information towards the periphery) will use the second variation (see below). The former type are shown in all figures with black arrow heads and the latter are shown with white arrow heads.

Consider a two region hierarchy (either figure 4.8(a) or figure 4.8(b)). If the lower region in the hierarchy generates an output (either due to sensor input or to cause a motor action), then the higher region may generate an arbitrary internal code to represent the action of the lower region. Having generated this internal code the higher region should learn so that it is more likely to recognise the same event in the future (afferent learning). If the lower region receives feedback from the higher region then it should associate this input with the current activation in the lower layer, as it is this activation which has given rise to the new input (efferent learning).

Consider one sensor and one motor region (figure 4.8(c)). The motor region causes an output, the action of which affects the environment to cause a corresponding sensory input. The sensor region updates its output in response to this input to produce an arbitrary internal code. The sensor region should also update its synaptic weights to make the internal code more likely to represent the same situation in the future (afferent learning). In order for the motor region to associate the sensory effects resulting from the activations in the motor region with those activations it needs to update its synapses to reinforce the connections between its current output and the new output of the sensor region (efferent learning). This will result in similar inputs from the sensor region being more likely to trigger the same response. However, in many situations action should be triggered by different sensory events than they give rise to. In this case the connections to the motor region would need to use afferent learning. Apical inputs (see below) can be used to determine if the previous input to the motor region is a useful trigger for the action produced, by causing only 'interesting' events (triggers for actions which have consistent consequences) to be learnt.

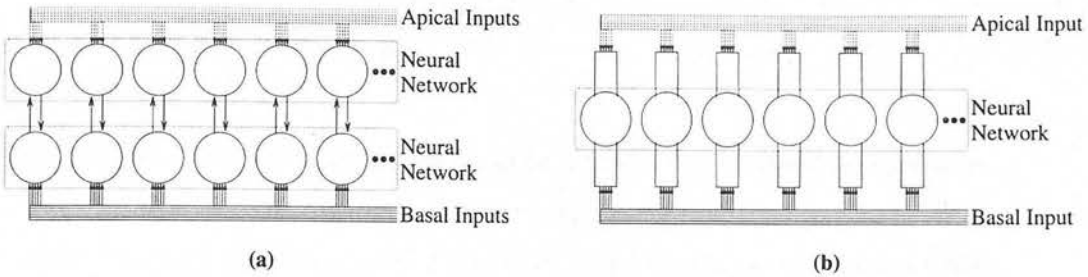


Figure 4.9: The architecture for the model cortical regions. A model cortical region consists of a neural network. However, this network is unusual in having two distinct sets of inputs. Both sets of inputs are processed in the same way, and the resulting activations influence learning at both sets of synapses. The algorithm can thus be interpreted as modelling interacting pairs of neurons (a). However, the preferred interpretation is that the algorithm represents a single layer of neurons with two distinct dendritic projections which process the inputs in the same way (b). In many applications there are no apical inputs and the model is then a standard neural network architecture. The neural network is two-dimensional, but is shown in one-dimension for simplicity.

4.2.4 Apical and Basal Dendrites

Each model cortical region has two separate sets of inputs. Each of these sets of inputs are processed in the same manner. The network could thus be considered to consist of two separate neural layers, with identical architectures (figure 4.9(a)), in which each layer receives distinct input projections and corresponding nodes in each layer interact. However, the preferred interpretation is that these processing units correspond to two distinct dendrites on the same neuron (figure 4.9(b)). These will be referred to as the apical and basal dendrites to correspond with the two dendritic arbors of cortical pyramidal cells (figure 3.3). At each iteration the activation of the apical dendrite is calculated for the current inputs, learning then takes place for synapses on both dendrites before the activation of the basal dendrite is calculated. The activation generated by each dendrite is calculated using the same procedure. The activation of each dendrite affects learning in the other but the output activation of a node is due to the activation of the basal dendrite only.

In many applications only the basal dendrites receive input (there are no apical inputs). In this case the neural network has a standard architecture. The output of the neural network is the activation caused by the inputs to the basal dendrites, and unsupervised learning occurs to make these activations find an appropriate representation of the input data. When apical inputs are present they have no direct effect on the output, but do affect learning on the basal dendrite, and similarly activation on the basal dendrite affects learning at synapses on the apical dendrite. Learning is then a tripartite function of pre-synaptic activity and the post-synaptic activation of both dendrites:

$$\Delta q_{ij}^{basal} = \beta lr(x_j) \otimes lr^{+}(y_i^{basal}) \otimes lr(y_i^{apical}). \tag{4.9}$$

$$\Delta q_{ij}^{apical} = \beta lr(x_j) \otimes lr^{+}(y_i^{apical}) \otimes lr(y_i^{basal}). \tag{4.10}$$

Such a modification to the learning scheme biases nodes to represent input patterns that are correlated

across the two input streams. The input to one dendrite can be used as a reinforcement or supervisory signal for learning at the other dendrite. Learning occurs after the input to the apical dendrite has been processed, but before the basal dendrite is updated. Inputs to the apical dendrite at the current iteration thus affect the learning of the previous input to the basal dendrite. Hence, in a situation in which the apical inputs represent effects caused by the output of the region the tripartite learning rule will bias nodes to represent events which have consistent effects. This has been found necessary to allow the learning of more complex behaviours (see section 6.2.2).

Other models have used multiple information streams to modulate one another's plasticity (*e.g.*, Becker, 1999; Der and Smyth, 1998; de Sa, 1994; Becker, 1996), or to modulate one another's activity and hence indirectly affect plasticity (*e.g.*, Kay and Phillips, 1994; Phillips et al., 1995; Phillips and Singer, 1997; Balkenius, 1995).

For pyramidal cells in the upper layers of the cortical sheet the basal dendrites predominantly receive input from feedforward projections, while apical dendrites predominantly receive input (in layer I) from feedback projections, from regions at a similar level in the hierarchy, and from the limbic system. (The limbic system is a widely projecting set of interconnected brain structures that may be concerned with emotion, along with other functions. Such a value system could have a role to play in reinforcement learning.) The input to layer I is considered to exert general controlling action upon cortical pyramidal cells and influence development (Mountcastle, 1998). The apical dendrites could, thus, be receiving appropriate information for the role suggested here. However, there is no biological support, nor known mechanism, for the idea that basal and apical dendrites modulate each other's plasticity. Another theory suggests a similar role for the apical inputs in modifying learning (Rolls and Treves, 1998). This theory differs in suggesting that the apical inputs may be used not only to affect learning but also to modulate output activations, for use in recall, semantic priming and attention. It is known that apical inputs, in isolation, can cause action-potentials (Mel, 1994; Mountcastle, 1998). By modulating activation dendrites could indirectly modulate each other's plasticity in a biologically plausible manner. However, without evidence that apical inputs do modulate learning at synapses on the basal dendrite or that modulatory signals are observed in cortical feedback pathways, the names apical and basal are nothing more than convenient labels which will be used to refer to two distinct types of input that have been found useful in this model.

Inputs to the basal dendrite are shown in all figures as solid lines and inputs to the apical dendrite are shown as dashed lines. The use of apical and basal dendrites and also afferent and efferent synapses means that there are four possible types of connection to a region. Figure 4.10 summarises the convention that will be used for showing these connection types in all figures.

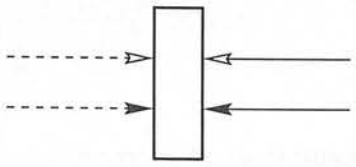


Figure 4.10: Key to the types of connection that may be received by a region. Solid lines represent inputs to the basal dendrite. Dashed lines represent inputs to the apical dendrite. Black arrow heads represent afferent synapses. White arrow heads represent efferent synapses.

4.3 Competition

Competition between nodes in a network is important to encourage nodes to come to represent different input patterns and so that multiple aspects of the input space can be represented. Habituation (section 4.3.1) ensures that nodes are uniformly active and lateral competition (section 4.3.2) ensures that they represent distinct inputs. This encourages nodes to represent the entire input space even though all nodes initially have identical synapses. Coverage of the input space is essential if the network is not to ignore some aspects of the data (Swindale, 1996). However, coverage is not guaranteed. To ensure coverage global information is required. Locally, a low activation in a node may be due to that input pattern being poorly represented or to other nodes being better representations. It might be possible to assess the level of lateral inhibition to distinguish these two cases and hence to ensure that poorly represented inputs are learnt. However, it is found that the algorithm is reasonably successful without such an additional mechanism.

Competition results in only a subset of nodes being active at any one time. In most algorithms the node which wins the competition will become the most active node in the network. However, in this algorithm that is not necessarily the case. It is thus necessary to identify two stages of competition. The first is a selection process in which a winning node is chosen based on the feedforward activation it receives. The second is a process of lateral inhibition in which the output activations of all nodes are calculated. The term 'winner' or 'winning node' will be used to refer to the node selected in the first process. The term 'most active node' will refer to the node with the highest activation after lateral inhibition.

4.3.1 Habituation

It is necessary to ensure that all nodes are given the opportunity to compete to represent inputs. Usually, nodes are all encouraged to be equally active so that the entire input space is represented with an even density of nodes. These methods presume that the input patterns occur with approximately equal probability and so constrain all nodes to be the winner of the competition with approximately equal frequency. This prevents any single node from repeatedly winning the competition and hence coming to

represent a disproportionately large region of the input space. Methods which achieve this aim include those which decrease (increase) an activation threshold for nodes which respond too infrequently (frequently) (Földiák, 1990; Intrator and Cooper, 1992; Rumelhart and Zipser, 1985) and methods which increase (decrease) the activation of nodes which respond too infrequently (frequently) (DeSieno, 1988; Ahalt et al., 1989, 1990; Ahalt and Fowler, 1993). Changing the threshold can be interpreted as neuronal growth (Balkenius, 1993): neurons receiving many signals grow and the enlarged soma subsequently requires more activation to make it discharge, while neurons receiving little input decrease in size and become easier to activate. Directly modifying the activation of a node can be interpreted as habituation, in which neural responses decrease in strength due to repeated stimulation. However, previous methods of modifying the activation of a node were intended to constrain nodes to win the competition with equal frequency rather than reflect the properties of habituation.

When using normalisation of the synaptic weights leaky learning can also be used to ensure that all nodes are responsive (Rumelhart and Zipser, 1985). Leaky learning allows nodes other than the winner to update their weights at each iteration. Eventually this will cause unresponsive nodes to move their RFs towards regions of the input space where there is input data, and hence allow them to compete to represent those inputs. However, when normalisation is not used, as here, such nodes would fail to compete successfully since they would have much smaller weights than other, more responsive, nodes.

Habituation has been used in the algorithm described here. Competitiveness is modified as a function of the number of times a node has been the winner of the competition. Nodes which have won the competition often have their ability to compete reduced ('habituated') by decreasing their activation, and the activation of nodes which seldom win is increased:

$$y_i = y_i \left(1 + \mu \left(I - nP_i\right)\right), \tag{4.11}$$

where I is the total number of iterations, P_i is the number of iterations for which node i was the winning node, n is the number of nodes in the network, and μ is a constant scale factor[18]. The degree of habituation applied to a node is a function of the difference between the number of times a node has won the competition and the expected number (given the number of interactions performed and the number of nodes in the network). The habituation factor changes at a constant rate independent of the total number of iterations that have taken place. This is in contrast to previous methods which take more and more iterations to cause the same degree of habituation as time progresses *i.e.*[19] $y_i = y_i + \mu\left(\frac{1}{n} - \frac{P_i}{I}\right)$

[18] The algorithm allows multiple nodes to be simultaneously active, however, there is only one winning node at each iteration and the value of P_i is updated for this one node. Habituation means that the winning node is not necessarily that which will have the highest output activation and in certain circumstances a node may win the competition with approximately the same frequency as all other nodes but never generate a strong output activation in comparison with other nodes.

[19] Both methods described here were originally formulated for use with networks in which competition was based on selection of the node with the smallest Euclidean distance between its synaptic weight values and the vector of input values. The equations

(DeSieno, 1988) and $y_i = y_i \frac{I}{P_i}$ (Ahalt et al., 1990, 1989; Ahalt and Fowler, 1993). The term used here has an effect more similar to habituation while still biasing nodes to win the competition an equal number of times in the long term.

The proposed algorithm thus uses nodes which have approximately equal firing rates and the firing strength is used to represent the degree to which an input pattern matches the RF of the node. An alternative encoding scheme is to use nodes which have unequal firing rates and which have firing strengths inversely proportional to the probability of the event represented (Barlow, 1990, 1994; Intrator, 1996). Unusual events are then indicated by high activation values which could be useful for detecting suspicious coincidences (Barlow, 1990, 1994; Intrator, 1996). However, it is not possible to use activation strength to encode both the probability and similarity of an input. Hence, this alternative scheme is not adopted as it is unable to represent the similarity between inputs and so cannot provide coarse coding. In addition, it provides no benefits since coincidences can still be detected between nodes with equal firing rates. Also, evidence suggests that neurons do attempt to keep a constant activation level (Quartz and Sejnowski, 1997; Ooyen, 1994).

Equiprobable firing rates are advantageous when representing a continuous input domain since the resolution of the representation will reflect the frequency of each feature. For a discontinuous input domain the use of equiprobable firing rates is more questionable since events are unlikely to occur with equal frequency (O'Reilly and McClelland, 1992). However, this constraint is not enforced too strongly, and seems necessary (in light of the argument in section 4.1.2) in order to allow more local encodings to be generated for events which occur frequently in order to learn appropriately abstract representations.

The selection of the winning node is also modified by noise. Early in learning, when synapses are weak and undifferentiated, this noise will have a strong influence on the selection of the winning node, but as the nodes come to represent different parts of the input space noise will have less effect and the strength of input activation (the match between the RF and the input pattern) becomes dominant in the choice of winner. The influence of noise early in learning is to provide symmetry breaking, so that initially identical nodes form different receptive fields. Noise also helps the formation of appropriate representations since it enables reorganisation of RFs before nodes become committed to representing particular inputs.

4.3.2 Lateral Inhibition

Lateral inhibition is an essential feature of many artificial, self-organising, neural networks. It provides a mechanism through which neurons compete to represent distinct patterns within the input space. Hence it makes the receptive fields, of individual nodes, more distinct (in the long-term) and the responses of

shown here are the equivalents for competition based on the selection of the node with the highest activation. Hence, these equations can be compared directly with equation 4.11.

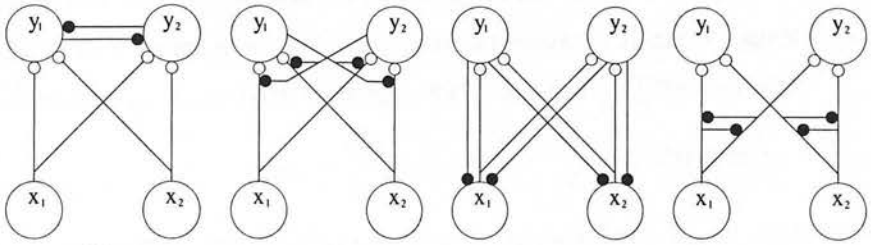


Figure 4.11: Models of lateral inhibition. Very simple neural networks consisting of two nodes which each receive two inputs are shown. Nodes are shown as large circles, excitatory synapses are shown as small open circles and inhibitory synapses are shown as small filled circles. (a) The standard model of lateral inhibition. (b) The pre-synaptic lateral inhibition model. (c) The negative feedback model. (d) The SONNET-2 model.

nodes more selective (in the short-term) (Somogyi and Martin, 1985; Adorján et al., 1998).

4.3.2.1 Standard Lateral Inhibition

The majority of self-organising neural networks use lateral inhibition to provide competition between the outputs of nodes (Földiák, 1989, 1990; Marshall, 1995a; Sirosh and Miikkulainen, 1994; Swindale, 1996; von der Malsburg, 1973), or produce the same effect without explicitly representing connections between nodes (Kohonen, 1997; Ritter et al., 1992; Rumelhart and Zipser, 1985). In this 'standard' architecture (as shown in figure 4.11(a)) lateral inhibition, via inhibitory synaptic weights (w_{ik}), occurs between the outputs of nodes to provide competition for the refinement of the feedforward weights. At equilibrium the output of each node will be:

$$y_i = \left(\sum_{j=1}^{m} (q_{ij} x_j) - \sum_{k=1(k \neq i)}^{n} (w_{ik} y_k) \right)^+ ,$$

where $(v)^+$ is the positive half-rectified value of v.

This standard architecture has been used (with variations to the learning rules) to find distributed, local, and topological representations. However, no single algorithm learns them all, due to conflicting requirements for lateral inhibition in forming local and distributed representations.

To learn a local representation the lateral weights simply need to become sufficiently strong to allow the activity of a single node to be dominant at any one time, while ensuring (*e.g.*, by use of habituation) that all nodes do respond to some inputs (to prevent a subset of nodes representing all the patterns). The process of finding the equilibrium state for the network requires iteration to find the final activations of all the nodes. This can be extremely computationally expensive (Harpur and Prager, 1994; Sirosh et al., 1996) and so is avoided in many neural network architectures in which a local code is generated as follows. Since lateral inhibition needs to be strong between all nodes the value may be fixed and individual connections need not be represented. Furthermore, the process of

competition between the nodes can be approximated by selecting the node with the highest feedforward activation at each iteration to be the winner of the competition and allowing this node to inhibit all the others. In this case the effects of strong lateral connections can be approximated by reducing all other activations (soft competition), or by setting the activations of all nodes, except the winner, to zero (hard, or winner-take-all, competition Kohonen, 1997; Rumelhart and Zipser, 1985):

$$y_{win} = \sum_{j=1}^{m} (q_{win,j} x_j), \qquad y_i = 0 \quad \forall i \neq win.$$

In this case no lateral connections need to be represented, however, this implementation still generates competition between node outputs as it provides competition for the right to be active. A topologically organised map may be formed by varying lateral inhibition strength as a function of the distance between nodes. Many topological map algorithms actually use a neighbourhood function to directly modify corresponding synapses on nearby nodes and use a winner-take-all mechanism to generate the output of the network (Kohonen, 1997; Ritter et al., 1992; Goodhill, 1993). Hence, such networks predict some other competitive mechanism, in addition to lateral inhibition, as these two functions have conflicting requirements which could not be implemented by lateral inhibition alone.

To learn a distributed code the lateral weights between individual nodes need to be explicitly represented as the strength of lateral inhibition will be specific to the patterns represented by those nodes. Thus, the lateral inhibition strength is not fixed but needs to be learnt as the feedforward weights are learnt. Lateral weights must be found that are sufficiently strong to prevent all nodes representing the same patterns, but that are sufficiently weak to allow more than one node to be active if multiple inputs are presented simultaneously. Thus, there is a balance to be struck between too little and too much competition. The activity of the nodes in response to the presentation of each input is ambiguous as to whether the weights have found the correct compromise: when two nodes are simultaneously active it can indicate either that lateral weights are too weak and have allowed both units to respond to the same input, or that each node has responded correctly to the simultaneous presentation of distinct inputs; when a single node is active it can indicate either that the lateral weights are too strong to allow other nodes to respond to other inputs, or that only one input pattern is present. To resolve this ambiguity it is assumed that each pair of input patterns occur with equal probability and the learning rules are designed to modify the lateral weights to ensure pairs of nodes are coactive with equal frequency, *e.g.*, $\Delta w_{ik} = \alpha y_k (y_i - w_{ik})$ (Marshall, 1995a), or $\Delta w_{ik} = \alpha \left(y_k y_i - p^2\right)$ (Földiák, 1990, 1992) (where p is the probability of each input pattern occurring, assumed *a priori*). A drawback with this assumption is that a representation will be unstable if there are more nodes than input patterns. In such a case nodes which do not represent an input pattern will receive weaker and weaker inhibition until they do become responsive after which they may become inhibited and unresponsive again, for a short time, or may take over representing one of the input patterns from another node (Marshall, 1995a). It is thus necessary to know beforehand how many patterns are present in the data. A more significant problem is that such

a network will modify its lateral weights to ensure that all pairs of representations are coactive with approximately equal frequency, whether or not this is appropriate. Hence, if certain pairs of stimuli never occur together in the input data then the network will adjust the lateral weights to force the representations of these mutually-exclusive pairs to be coactive and so will generate spurious representations. For example, in order for a network to represent the colour and shape of objects it should generate a factorial code in which nodes representing different colours and different shapes can be coactive. Using this form of lateral inhibition such a network will correctly respond to white squares and black triangles, but will also generate representations of black-white squares and black square-triangles even though such stimuli never exist in the input data. This results in an inaccurate representation. Hence, this form of lateral inhibition fails to provide factorial coding except for the exceptional case in which all pairs of patterns co-occur together. Furthermore, this approach is also incompatible with using lateral inhibition to form a topological representation, since modifying the strength of competition by a function of the distance between the nodes would upset the delicate balance in the strength of the lateral weights.

The standard architecture of lateral inhibition is restricted to representing mutually exclusive events or representing data in which all pairs of events occur simultaneously with equal frequency. The rules for learning the two forms of representation are incompatible. The rules for local coding cannot be used to form a distributed representation since they allow only one node to be active at a time. Alternatively, the rules for distributed coding cannot be used to represent mutually exclusive inputs since the lateral weights would weaken until each pair of nodes was occasionally coactive. This form of lateral inhibition thus fails to generate all the forms of place coding that are required. The restrictions on the type of data that can be represented and the incompatibility between algorithms for finding these representations is a fundamental limitation of the architecture used, not of any particular learning algorithm. Hence, modification to the architecture rather than the learning rules is required.

4.3.2.2 Pre-synaptic Lateral Inhibition

An architecture that uses lateral inhibition of pre-synaptic inputs (figure 4.11(b)), to provide competition for inputs rather than between outputs, overcomes the incompatibility between local and distributed coding. At equilibrium the output of each node will be:

$$y_i = \sum_{j=1}^{m} \left(q_{ij} x_j - \sum_{k=1(k\neq i)}^{n} (w_{ikj} y_k) \right)^+, \tag{4.12}$$

where w_{ikj} is the strength of the inhibitory synapse from node k to synapse j of node i.

If the lateral weights have somehow become selective in only inhibiting inputs to which the inhibiting node is most responsive, then each node will attempt to 'block' its preferred inputs from activating

other nodes. Nodes compete for the right to receive inputs rather than for the right to generate outputs. If two nodes try to represent the same input pattern there will be strong competition between them. If two nodes represent distinct patterns each node can respond to its preferred input without inhibiting the other node (assuming there is not much overlap between patterns). Thus the only constraint, for both local and distributed coding, is that lateral weights are sufficiently strong to allow nodes to claim their preferred input pattern. There is thus no incompatibility between the requirements for local and distributed coding, and an identical algorithm can find either as appropriate to the input data. In addition, since lateral weights can continue to increase, modifying them by a function of distance between nodes (to encourage the formation of a topologically ordered representation) does not prevent correct codes being found eventually. Any learning algorithm that refines the lateral weights to allow a node to strongly inhibit its preferred input pattern from reaching other nodes, while reducing its inhibition of other inputs, can generate distributed, local, and topological representations[20]. This ability is primarily a property of the architecture rather than any particular learning rules. Furthermore, since this method only needs to assume that the firing rates of individual nodes are equal, rather than coactivations of pairs of nodes, it is stable if there are more, or less, nodes than input patterns and if the input data contains pairs of patterns which are mutually exclusive.

One serious disadvantage of this architecture would seem to be the larger number of lateral weights required, however, these weights do not need to be individually learnt, resulting in this method being computationally efficient. The lateral weight w_{ikj} needs to inhibit the input j from reaching node i if the inhibiting node, k, is responsive to that input, *i.e.*, w_{ikj} needs to be strong if q_{kj} is strong. Thus, the lateral weights need simply to be a scaled version of the inhibiting node's feedforward weights, *i.e.*,

$$w_{ikj} = \frac{\alpha}{\beta} q_{kj},$$

and the activation function becomes:

$$y_i = \sum_{j=1}^{m} \left(q_{ij} x_j - \sum_{k=1(k \neq i)}^{n} \left(\frac{\alpha}{\beta} q_{kj} y_k \right) \right)^+ . \tag{4.13}$$

In a biologically plausible model the lateral weights could be represented and learnt separately, using a learning rule, such as; $\Delta w_{ikj} = \alpha f_3^{lateral}(x_j, y_k)$, while the feedforward weights were learnt, independently, using; $\Delta q_{kj} = \beta f_3^{forward}(x_j, y_k)$. The assumption is simply that the learning rule for the forward weights ($f_3^{forward}$) is equivalent to the learning rule for the lateral weights ($f_3^{lateral}$), and that both weights have the same initial value. This simplification reduces the number of weights that need to be learnt and represented by $n(n-1)$. Since synaptic weights are initially very small and since the

[20] It was found that normalised Hebbian learning, the outstar rule, and the covariance rule (see section 4.2.1) could all produce similar results to those given in chapter 5 when applied to finding simple maps (section 5.3) and solving the bars problem (section 5.1). In addition, results were fairly insensitive to the values of the learning parameters used.

activation of the inhibiting node is a function of its synaptic weights, in order for the lateral inhibition to be effective the scaling factor needs to be large (*i.e.*, $\alpha \gg \beta$).

Equation 4.13 represents the equilibrium activation for nodes after lateral inhibition. The interaction between nodes means that iteration will be required to find this equilibrium state, and hence this represents a dynamical implementation of the activation function. Using the standard method of lateral inhibition it was seen above that the process of competition between the nodes could be approximated by selecting a winning node to inhibit all others. When using pre-synaptic lateral inhibition the same reduction in computational complexity can be achieved since the output of the network can be found without numerical integration. Instead, a single winning node (win) can be selected to inhibit all the others at each iteration:

$$y_{win} = \sum_{j=1}^{m} \left(q_{win,j} x_j \right), \qquad y_i = \sum_{j=1}^{m} \left(q_{ij} x_j - \frac{\alpha}{\beta} q_{win,j} y_{win} \right)^+ \quad \forall i \neq win. \tag{4.14}$$

The inhibiting node is selected to be that which is most strongly activated by the current input, after habituation. This simplification succeeds since the inhibiting node competes with all others, forcing preferred inputs to become differentiated. However, since nodes which have learnt distinct input patterns do not inhibit each other it does not, necessarily, generate a local code and other nodes can continue to respond (and learn) even if they have not been selected as the winner. Section 5.2 points to some disadvantages of selecting a winning node based on feedforward activation only, however, for the vast majority of experiments this simplification is successful and hence the computational advantages outweigh these disadvantages.

The analysis given here suggests that a neural network architecture using pre-synaptic lateral inhibition has significant advantages over the standard model of lateral inhibition between node outputs. Such an architecture uses competition for inputs to nodes rather than competition to generate output activations. Unlike the standard model, this architecture allows a single learning algorithm to find distributed, local, and (with minor modification) topological representations of the input. It is thus not necessary to know *a priori* the structure of the data in order to generate a good representation for it. In addition, the representations are stable so that it is also not necessary to know *a priori* the number of nodes required to form the representation. Furthermore, these advantages are achieved with no increase in computational complexity over the standard architecture. These abilities are a function of the network architecture, rather than any clever learning rules.

The standard architecture requires excitatory cells to inhibit the outputs of other nodes. Such an architecture is anatomically and physiologically implausible since excitatory cells can only form excitatory synapses. This form of inhibition could be implemented between cortical cells via inhibitory

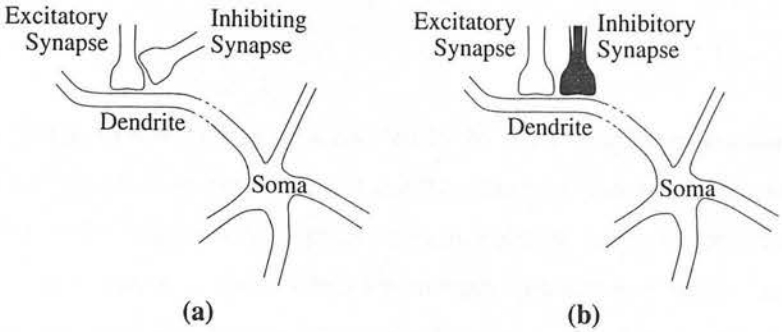


Figure 4.12: Mechanisms for implementing synapse specific inhibition. (a) Pre-synaptic inhibition through an axo-axonal synapse may prevent an input reaching the cell without the inhibiting synapse operating directly on the cell.(b) An inhibitory synapse near to an excitatory input can inhibit that input, while having no effect other branches of the dendritic tree.

interneurons providing either somatic inhibition or inhibition of the axon initial segment (Somogyi and Martin, 1985). All artificial neural network models of this kind make the same simplifying assumption that the inhibitory action of interneurons can be modelled using direct inhibitory connections between cells.

The proposed architecture requires inhibition specific to particular synaptic inputs. Such synapse specific inhibition might be implemented biologically either through pre-synaptic inhibition via an excitatory synapse on the axon terminal of the excitatory input (Bousher, 1970; Shepherd, 1990; Kandel et al., 1995) (figure 4.12(a)), or through an inhibitory synapse proximal to the excitatory input on the dendritic tree (Somogyi and Martin, 1985) (figure 4.12(b)). The former mechanisms enables an excitatory neuron to directly inhibit the input to another node without the action of an inhibitory interneuron. However, although axo-axonal, terminal-to-terminal, synapses are common in both vertebrate and invertebrate nervous systems (Brown, 1991; Kandel et al., 1995), there is no evidence for this mechanism of inhibition between cells in the neocortex (Mountcastle, 1998). Although the architecture has been described as providing pre-synaptic inhibition it could equally well be interpreted as a model of inhibitory synapses near to excitatory synapses along the dendritic tree. Since dendrites have nonlinear behaviour the effect of an inhibitory input will be greater at nearby synapses than at synapses on distant parts of the dendritic tree. This later mechanism relies, as does the standard architecture for lateral inhibition, on the action of inhibitory interneurons and the assumption must be made that the action of inhibitory interneurons can be approximated by using direct connections. However, there is also no support for this mechanism of inhibition in the neocortex since the majority of inhibitory synapses to pyramidal cells terminate on the soma or the axon initial segment rather than the dendrites. Hence, although this architecture is biologically plausible it is anatomically unjustified. However, at the computational level it does have some justification.

Although there would seem to be no simple biological mechanism for providing the form of lat-

eral inhibition required for this model it might be accounted for by more complex mechanisms. The diversity of forms of inhibitory cell in the cortex suggest that there are large variety of inhibitory functions (Somogyi and Martin, 1985; Houghton and Tipper, 1996). In addition, further mechanisms would be needed to account for the variation in lateral inhibition strength with distance required to form a topologically ordered representation. The lateral inhibition mechanisms providing competition between cortical neurons may well be different from those helping to form a topological organisation, although for simplicity both have been lumped together as a single inhibitory mechanism in this model.

4.3.2.3 Non-Standard Lateral Inhibition

It is obvious that inhibition could potentially act between several other points in a neural network, some of which have been explored in other non-standard architectures.

Negative Feedback Model. In this model (figure 4.11(c)) inhibitory feedback from each node inhibits the inputs to the entire network (Fyfe and Baddeley, 1995; Fyfe, 1997; Charles and Fyfe, 1998; Harpur and Prager, 1996). As with pre-synaptic lateral inhibition these weights could be learnt, but as a simplification are set to the same values as the corresponding feedforward weights. However, unlike the model proposed here, inhibition is to the inputs themselves and a node cannot entirely inhibit the input to all other nodes (without entirely inhibiting its own input). Since the inhibitory strength between individual nodes cannot vary there is no possibility of finding topological representations (without resorting to direct synaptic modifications on neighbouring nodes (Fyfe, 1996), or a second lateral inhibition mechanism). A similar architecture of lateral inhibition has been used by Sastry et al. (1999).

SONNET-2. In this model (figure 4.11(d)) inhibition is between the feedforward connections (Nigrin, 1993; Marshall and Gupta, 1998). The change in weight of the inhibitory synapses, and hence the strength of inhibition, is a function of the output activation of the inhibiting node and the input activations. In theory this architecture could develop local, distributed and topological representations, but it is far more complex than the pre-synaptic lateral inhibition model, and no such results have yet appeared.

Yuille's Winner-Take-All Model. In this model all the inputs to a node are inhibited equally and in proportion to the total output from all other nodes in the network (Yuille and Greynacz, 1989; Yuille and Geiger, 1995). At equilibrium the activation of each node will be:

$$y_i = \left(\sum_{j=1}^{m} q_{ij} x_j \right) e^{-\epsilon \sum_{k \neq i} y_k}.$$

This network was explicitly designed to form winner-take-all representations. There is no obvious way to modify this architecture to form distributed or topological representations.

4.3.2.4 Topology Preservation

By allowing the lateral inhibition strength to vary as a function of the distance between nodes it is possible to preserve the topology of the input space (Swindale, 1996; Sirosh and Miikkulainen, 1994). Unlike a neighbourhood function which directly modifies corresponding synapses on neighbouring neurons, lateral inhibition indirectly modifies synaptic weights by modifying node activations which in turn affects the synapses through an activity-dependent modification rule. The enforcement of the topology is thus weaker but it is sometimes an advantage not to force neighbouring nodes to have similar synaptic weights: in situations where the input space is discontinuous a neighbourhood function may force neighbouring nodes, representing inputs on opposite sides of the discontinuity, to modify their synapses to be more similar, which can result in nodes representing non-existent inputs in order to have similar RFs to their neighbours.

To modulate the strength of lateral connections as a function of distance the activation of a node becomes:

$$y_i = \sum_{j=1}^{m} \left(q_{ij} x_j + \frac{\alpha}{\beta} q_{win,j} y_{win} \Omega\left(win, i\right) \right)^{+} \quad \forall i \neq win \tag{4.15}$$

where $\Omega\left(win, i\right)$ is a function of the distance between the inhibited node i and the inhibiting node win. The minus sign in front of the second bracketed term has been absorbed into Ω.

A function which increases inhibition with distance needs to be used for Ω. Such a bias in connection strength will encourage nearby neurons to be active for similar input patterns, while the increasing competition between more distant neurons will result in them coming to represent dissimilar inputs. A simple Gaussian centred on the winning node; $\Omega(win, i) = \left(\exp \frac{\|\vec{z}_{win} - \vec{z}_i\|^2}{-2\sigma_I^2} - 1 \right)$, has been found sufficient to generate simple topological maps (Spratling and Hayes, 1998). However, by including a short range excitatory region the mapping has been found to be more uniform. Hence, the function used is: $\Omega(win, i) = \left(\exp \frac{\|\vec{z}_{win} - \vec{z}_i\|^2}{-2\sigma_I^2} + \exp \frac{\|\vec{z}_{win} - \vec{z}_i\|^2}{-2\sigma_E^2} - 1 \right)$, where $\vec{z}_i$ is the physical position of a node in the network (*i.e.*, its position in the neuronal array; not the position of its receptive field in the input space) and $\|\vec{z}_{win} - \vec{z}_i\|$ is the Euclidean distance between nodes win and i. This function provides mutual excitation over a short-range and longer-range inhibition which increases in strength with distance. This local region of excitatory lateral connections provides cooperation between nodes, while longer-range inhibition provides competition. Local excitation or weak local inhibition means that nodes in the neighbourhood of the winner remain active. The output of the network is thus the activity of a group of nodes centred around the winner. For a continuous input space this will be a topologically organised coarse code.

The major variable in experiments has been the σ_I and σ_E parameters which control the degree to which lateral weights were modulated as a function of distance between nodes. Two sets of values have been used.

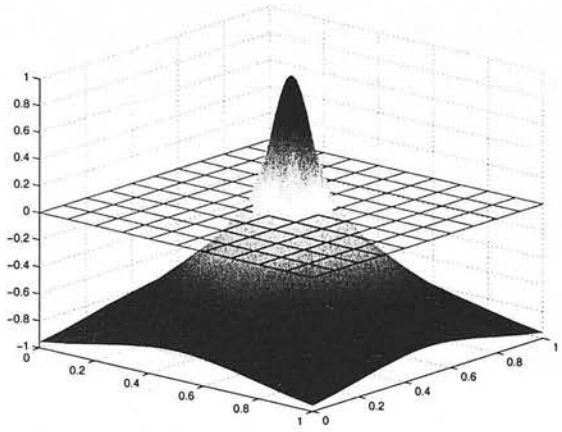


Figure 4.13: Scaling function for lateral inhibition. The shape of the function used to modulate the lateral inhibition strength in order to generate a topological map. The winning node is represented at coordinates (0.5,0.5). Local lateral weights are positive while more distant weights are inhibitory, with the strength of inhibition increasing with the physical distance from the winning node.

- For topological maps, and for coarse coding, $\sigma_I = \frac{1}{3}l$ (where l is the maximum distance between any two nodes in the region) and $\sigma_E = \frac{1}{3}\sigma_I$. The form of this function is illustrated in figure 4.13. The values of σ are scaled by the largest distance between any two nodes in the network in order to provide stability when using regions containing variable numbers of nodes.

- For local or distributed coding which is not topologically organised σ_I is small (*e.g.*, $\sigma_I = 0.01$) and $\sigma_E = 0$ so that all nodes inhibit each other equally, but each node has no self-excitation or inhibition (this is equivalent to equation 4.14).

4.4 Activation Rules

Each model neuron must calculate its response, or activation, given the current input. This activation is influenced by competitive mechanisms between the nodes (section 4.3): the activation is modified by habituation in order to select the winning node; and the activation is then modified by the effects of lateral inhibition. In addition, thresholding to remove low values of both input and output activation is applied in order to reduce noise and to make nodes more selective:

$$\text{if } x_j^{in} < \tfrac{1}{2}\tilde{x} \text{ then } x_j^{in} = 0; \tag{4.16}$$

$$\text{if } y_i^{in} < \tfrac{1}{2}\tilde{y}_i \text{ then } y_i^{in} = 0. \tag{4.17}$$

The threshold was chosen to be half the long-term trace by trial and error. This value is sufficiently high to be useful for removing noise without being too high to remove the signal. The algorithm was found to be insensitive over a large range of threshold values.

These mechanisms do not vary, but the contribution of the current input to the output activation is varied by using two distinct mathematical models for the neural units in a network. These differ in the combination function that is used: a linear, and a nonlinear combination function are considered. These functions affect the activation caused by synaptic inputs but have no effect on the subsequent competition between nodes or learning. The appropriate node type can thus be 'plugged in' to the algorithm described above.

4.4.1 Linear Units

The contribution of the current input to the activation of the node, before the application of the competitive mechanisms, is calculated as the sum of the input at each synapse multiplied by the weight of each synapse:

$$y_i = \sum_{j=1}^{m} q_{ij} x_j. \tag{4.18}$$

Hence, for the linear model each input to the node is independent. A linear model is sufficient for many applications, and will be used in the majority of cases in chapters 5 and 6.

Since the transfer function is a nonlinear threshold and since lateral inhibition uses a nonlinear selection process, the behaviour of these nodes is actually nonlinear. However, such nodes will be referred to as linear, due to their linear combination function.

4.4.2 Nonlinear Units

Certain applications of artificial neural networks require more powerful computational mechanisms than the standard linear threshold unit (a node using a linear combination function and a threshold transfer function). In addition, the response properties of biological neurons display nonlinearities, hence, higher-order units are also required to model such cells.

- To form a topological map of egocentric space (to guide reaching movements) for an agent with movable 'eyes' and 'neck' a node within the map would need to respond, exclusively, to all combinations of gaze direction and neck rotation which bring fixation to the same point in space. Cells in the parietal area of the neocortex respond to a preferred retinal location but are modulated, multiplicatively, by eye and head position (Anderson et al., 1985; Brotchie et al., 1995; Zipser and Andersen, 1988).

- To form an invariant representation (*e.g.*, of an object under visual transformations) requires that sets of input patterns be associated together while being distinguished from other, possibly overlapping, patterns. Cells in the inferotemporal area of the neocortex are selective to the form of visual stimulus but insensitive to location and scale of the image (Tovee et al., 1994). Complex

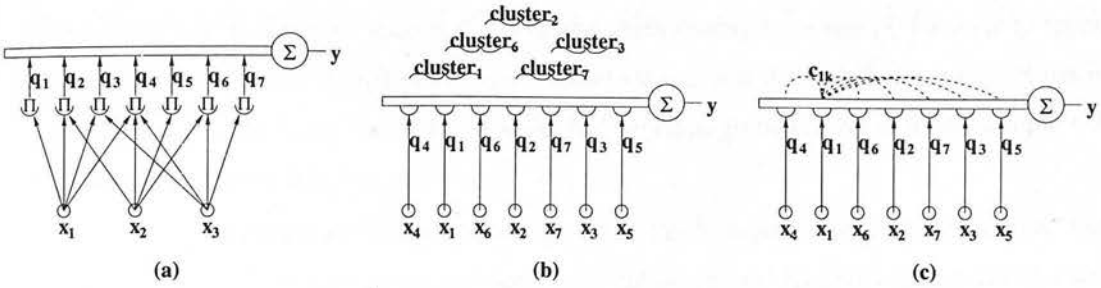


Figure 4.14: Models of nonlinear neurons. (a) A sigma-pi neuron. (b) A clusteron neuron. (c) The proposed model.

cells in the primary visual cortex respond to appropriately orientated line segments located anywhere within a receptive field (Edelman et al., 1997).

Hence, many neural models with nonlinear response properties have been proposed (Salinas and Abbott, 1996; Rumelhart et al., 1986; Mel and Koch, 1990; Pouget and Sejnowski, 1995; Mel, 1992, 1994; Feldman and Ballard, 1982; Durbin and Rumelhart, 1989; Güler and Sahin, 1994; Sheperd and Brayton, 1987; Salinas and Abbott, 1997b,a).

There is evidence to suggest that information processing takes place in the dendritic trees of biological neurons (see Mel, 1994, for a review). Local thresholding of the summed input from clusters of excitatory synapses within the dendritic tree could account for the multiplicative responses of cells (Feldman and Ballard, 1982; Mel, 1990b, 1999). A single such unit is thus as powerful as a multi-layer network of linear threshold units, and the outputs of many nonlinear units can be combined together to act as a larger, 'virtual', nonlinear unit (Mel, 1990b). For a nonlinear unit inputs to the dendrite are not independent. The nonlinearity can be considered to be due to processes occurring within the dendritic tree.

4.4.2.1 Sigma-Pi Units

The principal model of a neuron using a nonlinear combination function is the sigma-pi unit (figure 4.14(a)), in which the output activation (y) is calculated as the weighted sum of the products of independent sets, or clusters, of input values (x_k) (Rumelhart et al., 1986; Mel, 1990b):

$$y_i = \sum_{j=1}^{m} \left(q_{ij} \prod_{k \in cluster_j} x_k \right).$$

Hebbian learning, between the cluster's product and the unit's (required) output, can be used to find appropriate values for the synaptic weights (q) (Mel, 1990b; Mel and Koch, 1990). The inputs which form a cluster need to be predefined: either all clusters are defined to be the same size, in which case all combinations of m' inputs are used (this is unsatisfactory as the appropriate cluster size may not be

known before-hand, nor may all the required clusters be of the same size); alternatively all combinations of $\leq m'$ inputs are used (which is computationally prohibitive, due to the number of clusters required, if the number of inputs or the maximum cluster size is large). Even using the more tractable first approach the number of parameters required can easily be unreasonably large (*e.g.*, for a node to represent all clusters of size 5 from a total of 50 inputs would require over 2.1×10^6 synapses). Noise is also a problem due to the input values being multiplied together, so that noise in a single input value will affect the activation of the entire cluster.

To overcome these problems individual clusters can be represented as 'product units' which can learn arbitrary polynomial functions of inputs via gradient descent (Durbin and Rumelhart, 1989). Each product unit thus learns which inputs are relevant to the cluster it represents. However, it is then necessary to use a multi-layer network of both summing and product units as well as using supervised learning, and noise remains a problem.

4.4.2.2 Clusteron Units

The clusteron (figure 4.14(b)) is a more biologically inspired attempt to model dendritic cluster sensitivity (Mel, 1992, 1994). The output of a unit is calculated as:

$$y_i = \sum_{j=1}^{m} \left(q_{ij} x_j \sum_{k=j-m''}^{j+m''} q_{ik} x_k \right).$$

The contribution of a synapse j is thus affected by the activity of neighbouring synapses (within radius m'' of j) on the one-dimensional dendrite. A 'cluster' is thus defined differently in this case. In contrast to sigma-pi units, where each set of inputs has a synaptic weight, in the clusteron each input has a synaptic weight and the term 'cluster' is used to refer to those inputs which can affect the activation received by a particular synapse. All clusters are of a constant, predefined, size and hence suffer from the same problems as a sigma-pi unit with the same restriction (see above), with the additional problem that false clusters are formed (*e.g.*, in figure 4.14(b), having clustered synapses 1 and 6 and synapses 6 and 2, synapses 1 and 2 will also be in a cluster, whether or not this is justified by the input data). Noise in one input will affect the contributions of all the other synapses in the cluster so that the effect of noise is almost as strong as in a sigma-pi unit. Moreover, because there is no way of isolating a cluster from the activations of other synapses in the neighbourhood there is crosstalk and the clusteron is inherently much more noisy. The effect of neighbouring synapses is not conjunctive[21], and restricting the dendrite to one-dimension limits the number of clusters any one synapse can participate in.

Clustering is determined by the ordering of the synapses along the dendrite, and is formed by randomly swapping the positions of those synapses that have been least involved in firing the unit; this is

[21] If the input activation to all synapses in a cluster, other than synapse j, is zero then the activation at synapse j is $q_{ij}^2 x_j^2$. Synapses enhance the activation at other synapses rather than gating them.

little better than random search. The apparent huge reduction in the number of parameters in comparison to a sigma-pi unit (*e.g.*, a node with 50 inputs requires just 50 synapses) is due to the clustering of the inputs not being explicitly represented and is paid for in the search time required before clusters are formed. The probability of finding the correct clusters becomes small very rapidly with problem size (*e.g.*, the probability that within any random ordering of 50 synapses a specific cluster of 5 inputs is present is less than 2.2×10^{-5}).

4.4.2.3 Cluster Learning Units

The drawbacks of the sigma-pi unit, discussed above, are all practical issues. It is too computationally expensive to implement a unit with all possible clusters, while using a more tractable subset of clusters requires *a priori* knowledge and presumes clusters will all be of the same size. (Since the clusteron was designed to explore the effects of cluster sensitivity in models of the dendritic trees of biological neurons it does not consider these practical issues, and suffers from similar limitations.) A solution would be to learn the clustering, assuming that the learning procedure itself was tractable. A minimum set of clusters could then be found and the size of each cluster could be independent.

This section introduces a model of dendritic computation which learns clustering (figure 4.14(c)). In this model any synapse (k) may form (part of) a cluster with any other synapse (j). The degree to which both are in the same cluster is designated by c_{ijk} (the 'cluster weight'). The definition of 'cluster' used here is similar to that used in the clusteron; each input has an individual synaptic weight and other inputs in the cluster can affect the activation received by that synapse. The definition differs in that the set of inputs which form a cluster is determined by the strength of the cluster weights rather than by neighbourhood. For an individual synapse, j, all other inputs, k, which can affect the activation at synapse j are its cluster, these will be inputs for which the cluster weight $c_{ijk} > \kappa$ (see below, where κ is a constant threshold). The cluster weights define global pairwise interactions between synapses, however the input at synapse j can be affected by all other inputs for which $c_{ijk} > \kappa$ so that the cluster can be any size.

No *a priori* assumptions are made about the required clustering and so all cluster weights are initially zero ($c_{ijk} = 0 \ \forall i, j, k$). In this condition it is necessary that the node act as a standard linear neuron. As clusters are learnt there should be a gradual transition to nonlinear behaviour with the degree of nonlinearity increasing in proportion to the cluster weight (*i.e.*, as c_{ijk} increases the confidence that synapse k is part of a cluster with synapse j increases and the value of x_k should have increasing effect on the input at synapse j). An activation function which satisfies these requirements is:

$$y_i = \sum_{j=1}^{m} q_{ij} \min\{\frac{x_1 + \kappa}{c_{ij1}}, \frac{x_2 + \kappa}{c_{ij2}}, \ldots \frac{x_{j-1} + \kappa}{c_{ij,j-1}}, x_j, \frac{x_{j+1} + \kappa}{c_{ij,j+1}}, \ldots \frac{x_m + \kappa}{c_{ijm}}\}. \tag{4.19}$$

Such a node will act as a linear unit until $c_{ijk} > \kappa$ for some connection. Once this condition has been

reached the input to synapse j can be reduced by low activity in x_k or 'turned off' if input x_k is inactive. In all other situations the activation of the node is simply the (linear) weighted sum of the input values[22].

In the worst case (when $c_{ijk} = 1 \; \forall k \in cluster_j$) the effect of noise is as bad as for a sigma-pi unit. However, the cluster weights are generally small, so that only much reduced values of x_k will affect the activation of synapse j. Thus input noise on any one source will only affect the activation at that synapse and not the activation received by other synapses in the cluster resulting in a much smaller effect on the output activation.

While it would appear that there are a large number of cluster weights to be learnt the number of parameters required is much less than for a sigma-pi unit (*e.g.*, with 50 inputs a node requires 50 synapses plus 2450 cluster weights) with the added benefit that there are no restrictions on cluster size. However, because each synapse has only one cluster weight vector, it can only participate in a single cluster. To represent patterns that require an input to participate in multiple clusters, a 'virtual' unit consisting of multiple competing nodes can be used. The individual nodes within a virtual unit use unsupervised competitive learning to divide the problem between them.

The explicit representation of clusters allows them to be formed without search. Since inputs likely to be part of the same cluster will be frequently coactive the clustering weights (c) and the synaptic weights (q) are learnt simultaneously (both sets of weights have zero strength initially):

$$\Delta q_{ij} \propto lr^+(x_j) \otimes lr^+(y_i), \tag{4.20}$$

$$\Delta c_{ijk} \propto lr(x_j) \otimes lr(y_i) \otimes lr(x_k). \tag{4.21}$$

The value of c_{ijk} is constrained not to change sign (*i.e.*, $c_{ijk} = (c_{ijk})^+$). Synaptic weight change is a function of the pre- and post-synaptic activity at that synapse. Cluster weight change is a function of the post-synaptic activity and the pre-synaptic activity at both synapses connected by that cluster weight. Terms in the learning rules have different limits on the values they can take for cluster weight learning and for synaptic weight learning; synaptic weight changes are constrained to be positive, while cluster weight changes can be either positive or negative. Hence, cluster weights can decrease as well as increase resulting in the formation of cluster weights being more conservative than the formation of synaptic weights. In addition, c for any synapse which has zero synaptic weight (*i.e.*, $q_{ij} = 0$) is reset so that $c_{ijk} = 0 \; \forall k$. Thus, cluster weights are smaller than synaptic weights and only inputs which form strong synaptic weights can form clusters with other inputs. This prevents false clusters being found and allows the node to realign its receptive field if the input distribution changes.

When using nonlinear nodes the learning rules for synaptic weights are modified slightly to restrict the pre-synaptic term to be positive or zero. This is necessary to allow a node to represent multiple sets

[22] Division by zero, when $c_{ijk} = 0$ at the start, is avoided by defining $\left(\frac{x_j+\kappa}{0}\right) = 1$ (an arbitrary finite value greater than the maximum input activation).

of co-occurring features, *i.e.*, to form multiple sets of clusters representing different input patterns. If this restriction was not used then when the node was activated by one set of inputs it would reduce its weights to inputs forming all other clusters. Although this solution works it is not ideal. It would be preferable to allow separate clusters in which the synaptic weights could be reduced without affecting the synaptic weights in other clusters. This would require the formation of multiple connections from each input to each separate cluster. Pre-synaptic cells in the neocortex are observed to form more than one synapse to the same post-synaptic cell (Ebdon, 1996), however, a mechanism that allowed multiple, separate, clusters to be learnt would need to be far more complicated than the current implementation.

In this model an initially standard linear node uses unsupervised learning to find clusters of inputs within which inactivity at one synapse can occlude the activity at the other synapses. Learning clusters of inputs, for nonlinear dendritic processing, provides a solution to some of the practical limitations of other models. The resulting neurons are no more powerful than other models with nonlinear input functions (nor more powerful than a multi-layer network of linear threshold units). However, this method does have the advantage of being more tractable than other implementations of nonlinear neurons. It provides unsupervised, and efficient, learning of synaptic clusters, without the prespecification of cluster size; allowing clusters to be of independent and arbitrary size. False clusters are not formed. Inactive synapses can block the activity of all other synapses in the cluster, but otherwise clustering has no effect on the activity. This allows the method to work with coarse coded inputs, and makes it relatively insensitive to noise in the input values.

While the spatial ordering of the synapses within the dendritic tree may be critical for a biological neuron, it does not mean that an artificial neuron must also use the ordering of the synapses to represent clustering. Instead, this implementation has used 'weights' to represent the degree of clustering between synapses. All sense of locality is thus lost, and this model does not attempt to represent local interactions within a fixed dendritic tree. In this sense it is not a biologically plausible model of nonlinear dendritic processing. However, for the static ordering of the synapses on a dendritic tree to be formed it must be learnt. The clustering weights of this model might therefore be interpreted as representing the mutual attraction of axons innervating a three-dimensional dendritic tree. Hence, while the learning rule for the synaptic weights might be interpreted as determining the existence of a connection, the learning rule for the cluster weights might be interpreted as determining the position of that connection.

4.5 Summary

This chapter has described the algorithm that has been implemented as a model of development. The reader should refer to tables 4.4, 4.5, 4.6, and 4.7 for a summary of the algorithm that has been described in the preceding sections. Table 4.8 gives details of the simplified algorithm which is used in most

applications. The section labels along the right-hand edge of these tables indicate where those steps in the algorithm are described within the main text.

Initialisation:

- $q_{ij} = 0\ \forall j, i$ where q_{ij} is the synaptic weight from input j to node i.
- $c_{ijk} = 0\ \forall i, j, k$ where c_{ijk} is the cluster weight from synapse k to synapse j in node i.
- $\tilde{x} = 0$ where $\tilde{x}$ is the long-term average, or time trace, of mean past activity for all inputs x_j.
- $\tilde{y}_i = 0\ \forall i$ where $\tilde{y}_i$ is the long-term average, or time trace, of past output activity for node i.
- $P_i = 0\ \forall i$ where P_i is the number of iterations for which node i was the winning node.
- $I = 0$ where I is the total number of iterations.

Parameter values:

- $\alpha = 0.0009$ where α is the learning rate for lateral connections.
- $\beta = 0.00000225$ where β is the learning rate for feedforward connections.
- $\sigma_I = \frac{1}{3}l$ (for topological coding), $\sigma_I = 0.01l$ (for non-topological coding) where σ_I is the radius of lateral inhibition.
- $\sigma_E = \frac{1}{3}\sigma_I$ (for topological coding), $\sigma_E = 0.0$ (for non-topological coding) where σ_E is the radius of lateral excitation.
- $\nu = 0.005$ where ν is the noise scale factor.
- $\mu = 0.001$ where μ is the habituation scale factor.
- $\lambda_x = 0.5$ (for temporal data), $\lambda_x = 1.0$ (for atemporal data) where λ_x is the sensitisation scale factor.
- $\tau_y = \tau_x = 0.001$ where τ_y and τ_x are the constants for calculating the time traces of the output and input respectively.
- κ *variable* where κ is the threshold for applying nonlinear dendritic processing.

Nomenclature:

- n is the total number of nodes in the region.
- m is the total number of inputs to a node.
- l is the maximum distance between any two nodes in the region.
- z_i is the physical location of node i in the network.
- $\hat{v} = \mathtt{max}_k\{v_k\}$ is the maximum value of any variable v for the current iteration.
- $\bar{v} = \mathtt{mean}_k\{v_k\}$ is the mean value of any variable v for the current iteration.
- $(v)^+$ is the positive half-rectified value of v.
- $\otimes$ is multiplication when either or both terms are positive and gives zero otherwise.

Table 4.4: Details of the proposed learning algorithm. Parameters and variables.

1. Calculate the activation of the apical dendrite (as described in table 4.6).
2. Do learning for the apical dendrite (as described in table 4.7).
3. Do learning for the basal dendrite (as described in table 4.7).
4. Calculate the activation of the basal dendrite (as described in table 4.6).
   Use the basal dendrite activations as the output of the region.
5. Continue from step 1.

Table 4.5: The details of the proposed learning algorithm. The main loop.

1. For each input, j:

   Read the new activation value x_j^{in}.

   Threshold low values such that if $x_j^{in} < \frac{1}{2}\tilde{x}$ then $x_j^{in} = 0$. (sec 4.4)

   Sensitise the input activation to previous activations, $x_j = \lambda_x x_j^{in} + (1 - \lambda_x)x_j$. (sec 4.2.2)

   Update the trace of the input activity, $\tilde{x} = \tau_x \bar{x} + (1 - \tau_x)\tilde{x}$ (only if $\hat{x} > 0$). (sec 4.2.1)

   Calculate the pre-synaptic activation, x'_{ij}, to node i from input j,

   for linear units, $x'_{ij} = x_j$,
   for nonlinear units, $x'_{ij} = \mathtt{min_k}\{\frac{x_1+\kappa}{c_{ij1}}, \frac{x_2+\kappa}{c_{ij2}}, \ldots \frac{x_{j-1}+\kappa}{c_{ij,j-1}}, x_j, \frac{x_{j+1}+\kappa}{c_{ij,j+1}}, \ldots \frac{x_m+\kappa}{c_{ijm}}\}$. (sec 4.4.2.3)

For each node, i:

   Calculate the activation due to the current input,

   $y_i^{in} = \sum_{j=1}^{m} q_{ij} x'_{ij}$.

   Threshold low values such that if $y_i^{in} < \frac{1}{2}\tilde{y}_i$ then $y_i^{in} = 0$. (sec 4.4)

   Calculate the activation, y_i^{bid}, modified by habituation and noise,

   $y_i^{bid} = y_i^{in}\,(1 + \mu\,(I - nP_i)) + \mathtt{rand}(\nu)\hat{y}^{in}$ (sec 4.3.1)
   where: $\mathtt{rand}(\nu)$ is a uniformly distributed random variable between $-\nu$ and $+\nu$.

Choose the winning node, win, such that $y_{win}^{bid} > y_i^{bid}$, $\forall i \neq win$. (sec 4.3.2.2)

   Increment I and P_{win} (only if $\hat{y}^{in} > 0$).

For each node, i:

   Calculate the output activation, after lateral inhibition,

   $y_i^{out} = \sum_{j=1}^{m} \left(q_{ij}x'_j + \frac{\alpha}{m\beta} q_{win,j} y_{win}^{in} \Omega(win, i)\right)^+$ (only if $\hat{y}^{in} > 0$), (sec 4.3.2.2)

   $y_i^{out} = 0$ (otherwise),

   where: $\Omega(win, i) = \left(\exp \frac{\|\vec{z}_{win} - \vec{z}_i\|^2}{-2\sigma_I^2} + \exp \frac{\|\vec{z}_{win} - \vec{z}_i\|^2}{-2\sigma_E^2} - 1\right)$ scales the lateral inhibition as a function of the physical distance between the nodes. (sec 4.3.2.4)

   Update the trace of the output activity, $\tilde{y}_i = \tau_y y_i^{out} + (1 - \tau_y)\,\tilde{y}_i$ (only if $\hat{y}^{in} > 0$). (sec 4.2.1)

Table 4.6: The details of the proposed learning algorithm. Activation.

Modify synaptic weights for both dendrites: (sec 4.2.1 and sec 4.2.4)

for linear nodes, $\Delta q_{ij} = \beta\; lr(x_j) \otimes lr^+(y_i^{out}) \otimes lr(y_{i,other\ dendrite}^{out})$.

for nonlinear nodes, $\Delta q_{ij} = \beta\; lr^+(x_j) \otimes lr^+(y_i^{out}) \otimes lr(y_{i,other\ dendrite}^{out})$. (sec 4.4.2.3)

Modify the cluster weights for both dendrites: (sec 4.4.2.3)

for nonlinear nodes only, $\Delta c_{ijk} = \beta\; lr(x_j) \otimes lr(y_i^{out}) \otimes lr(x_k) \otimes lr(y_{i,other\ dendrite}^{out})$.

Where: $lr(x_k) = \frac{m\left(\frac{x_k}{\bar{x}} - 1\right)}{\sum_{j=1}^{m} \left|\frac{x_j}{\bar{x}} - 1\right|}$, $lr(y_k) = \frac{n\left(\frac{y_k}{\bar{y}_k} - 1\right)}{\sum_{i=1}^{n} \left|\frac{y_i}{\bar{y}_i} - 1\right|}$, $lr^+(y_k) = \frac{n\left(\frac{y_k}{\bar{y}_k} - 1\right)^+}{\sum_{i=1}^{n} \left(\frac{y_i}{\bar{y}_i} - 1\right)^+}$.

(The post-synaptic term is rescaled when no node has a positive learning rate by using $\bar{y}$ as the threshold rather than $\tilde{y}_i$.)

Limit the maximum activation: $q_{ij} = \frac{q_{ij}}{\mathtt{max}\left(1, y_i^{out}\right)}$. (sec 4.2.1)

Prevent sign changes: $q_{ij} = (q_{ij})^+$, $c_{ijk} = (c_{ijk})^+$. (sec 4.2.1)

Table 4.7: The details of the proposed learning algorithm. Learning.

Initialisation:

$q_{ij} = 0\ \forall j, i$ where q_{ij} is the synaptic weight from input j to node i.
$\tilde{x} = 0$ where $\tilde{x}$ is the long-term average, or time trace, of mean past activity for all inputs x_j.
$\tilde{y}_i = 0\ \forall i$ where $\tilde{y}_i$ is the long-term average, or time trace, of past activity for node i.
$P_i = 0\ \forall i$ where P_i is the number of iterations for which node i was the winning node.
$I = 0$ where I is the total number of iterations.

1. For each input, j:

   Read the new activation value x_j^{in}.

   Threshold low values such that if $x_j^{in} < \frac{1}{2}\tilde{x}$ then $x_j^{in} = 0$. (sec 4.4)

   Update the trace of the input activity, $\tilde{x} = \tau_x \bar{x} + (1 - \tau_x)\tilde{x}$ (only if $\hat{x} > 0$). (sec 4.2.1)

2. For each node, i:

   Calculate the activation due to the current input,

   $y_i^{in} = \sum_{j=1}^{m} q_{ij} x_j$.

   Threshold low values such that if $y_i^{in} < \frac{1}{2}\tilde{y}_i$ then $y_i^{in} = 0$. (sec 4.4)

   Calculate the activation, y_i^{bid}, modified by habituation and noise,

   $y_i^{bid} = y_i^{in}\left(1 + \mu\left(I - nP_i\right)\right) + \texttt{rand}(\nu)\hat{y}^{in}$, (sec 4.3.1)
   where: $\texttt{rand}(\nu)$ is a uniformly distributed random variable between $-\nu$ and $+\nu$.

3. Choose the winning node, win, such that $y_{win}^{bid} > y_i^{bid},\ \forall i \neq win$. (sec 4.3.2.2)

   Increment I and P_{win} (only if $\hat{y}^{in} > 0$).

4. For each node, i:

   Calculate the output activation, after lateral inhibition,

   $y_i^{out} = \sum_{j=1}^{m}\left(q_{ij}x_j + \frac{\alpha}{m\beta} q_{win,j} y_{win}^{in} \Omega(win, i)\right)^+$ (only if $\hat{y}^{in} > 0$), (sec 4.3.2.2)

   $y_i^{out} = 0$ (otherwise),

   where: $\Omega(win, i) = \left(\exp\frac{\|\vec{z}_{win} - \vec{z}_i\|^2}{-2\sigma_I^2} + \exp\frac{\|\vec{z}_{win} - \vec{z}_i\|^2}{-2\sigma_E^2} - 1\right)$ scales the lateral inhibition as a function of the physical distance between the nodes. (sec 4.3.2.4)

   Update the trace of the output activity, $\tilde{y}_i = \tau_y y_i^{out} + (1 - \tau_y)\,\tilde{y}_i$ (only if $\hat{y}^{in} > 0$). (sec 4.2.1)

5. For each input, j, modify synaptic weights: (sec 4.2.1 and sec 4.2.4)

   $$\Delta q_{ij} = \beta\ \frac{m\left(\frac{x_j}{\bar{x}} - 1\right)}{\sum_{k=1}^{m}\left|\left(\frac{x_k}{\bar{x}} - 1\right)\right|}\ \frac{n\left(\frac{y_i^{out}}{\tilde{y}_i} - 1\right)^+}{\sum_{k=1}^{n}\left(\frac{y_k}{\tilde{y}_k} - 1\right)^+}.$$ (sec 4.2.1)

   Limit the maximum activation: $q_{ij} = \frac{q_{ij}}{\max\left(1, y_i^{out}\right)}$. (sec 4.2.1)

   Prevent sign changes: $q_{ij} = \left(q_{ij}\right)^+$. (sec 4.2.1)

6. Continue from step 1.

Table 4.8: The simplified algorithm for standard conditions. When used with linear units, inputs to the basal dendrite only, and afferent learning the algorithm can be written in the simplified form above.

Chapter 5

INTRA-REGIONAL DEVELOPMENT

This chapter will be concerned with learning in a single model cortical region. Such a region constitutes the simplest of assemblies, examples of which are shown in figure 5.1. The assemblies shown in figures 5.1(a) and 5.1(b) are equivalent; since arrows represent multiple inputs both networks are shown receiving information from multiple inputs. However, in certain circumstances it will be convenient to distinguish two (or more) separate 'streams' of inputs (as in figure 5.1(b)), specifically when these input streams come from distinct sources. All results are generated using single networks of neurons with afferent synapses and in most cases inputs to the basal dendrite only (except section 5.6 where the effect of connections to the apical dendrite is considered) and using linear nodes (except sections 5.4 onwards when nonlinear nodes will be used). Hence, for the most part the simplified algorithm given in table 4.8 applies.

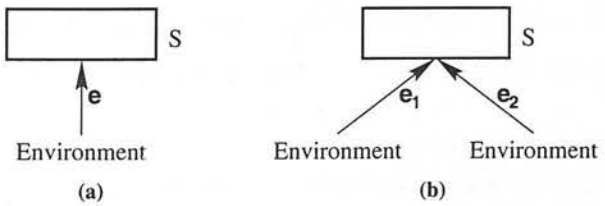


Figure 5.1: The assemblies used to test simple sensory abstraction and invariance.

These results explore the performance of the algorithm introduced in the previous chapter in learning useful representations of input distributions. It attempts to show that representations can be learnt that appropriately encode various input distributions and that these encodings generate accurate responses to the input. Some assessment is made of 1) how robust the algorithm is at finding such ap-

propriate representations, and 2) how stable the algorithm is in generating correct responses once the representations are formed.

5.1 Distributed and Local Coding

This section considers the formation of representations in which each node represents a single, distinct, feature of the input space. Nodes inhibit each other with equal strength ($\sigma_I = 0.01, \sigma_E = 0.0$).

A widely used test for factorial coding is the bars data set (Földiák, 1990; Harpur and Prager, 1996; Fyfe, 1997; Charles and Fyfe, 1998, 1997; Hinton et al., 1995; Hinton and Ghahramani, 1997). Input consists of an 8x8 grid on which each of the 16 possible horizontal and vertical bars are active with probability $\frac{1}{8}$. Typical examples of input patterns are shown in figure 5.2(a)[1]. When a network of 16 nodes was trained on this data it found an encoding in which each node represented exactly one bar and each bar was represented exactly once (such an encoding will be referred to as a 'suitable' representation of this data). This was determined by evaluating the preferred input of each node using the two forms of decoding discussed in section 4.1.3: decoding the synaptic weights, and recording the response. It was found that the synaptic weights corresponding to inputs from the preferred stimulus became much stronger than connection strengths from all other grid points (figure 5.2(b)), so that based on weight decoding a suitable representation was formed within 1600 training cycles. As the receptive fields of each node became selective to a particular stimulus so the response of nodes became specific to the presence of the preferred bar in the input data (figure 5.2(c)). When tested with a further 500 patterns, after training for 4000 iterations, the response generated by this network (for input data consisting of n bars) was such that the n most active nodes were those which correctly represented all the active bars in 76 out of 76 (100% of) unseen[2] (as well as 424 out of 424 previously seen) test patterns. A more reasonable test is to assume that the number of bars in the input pattern is unknown and to consider all those nodes that have an activation exceeding a predefined threshold. The value chosen for the threshold, by trial-and-error, was the sum of the mean activation of nodes to the current input and of the mean activation of nodes throughout the testing ($\tilde{y} + \bar{y}$). The same threshold was used for all tests and so its value was not optimised for individual experiments to maximise the discrimination between active and inactive nodes. Using this criterion, after 4000 training cycles, the active nodes correctly represent all the active bars in the input pattern (and only those bars) for 499 out of 500 (99.8% of) test patterns.

[1] A factorial coding of this input distribution could be found using an algorithm which applies the standard model of lateral inhibition (see section 4.3.2.1) since subfeatures all co-occur with equal frequency.

[2] Unseen patterns consist of previously unencountered combinations of bars: all individual bars are presented during training. In this case only 76 bar combinations out of the 500 test cases had not been previously presented to the network during training. Since unseen combinations of bars will tend to be those consisting of larger numbers of active bars the unseen test cases are biased towards the harder classification problems.

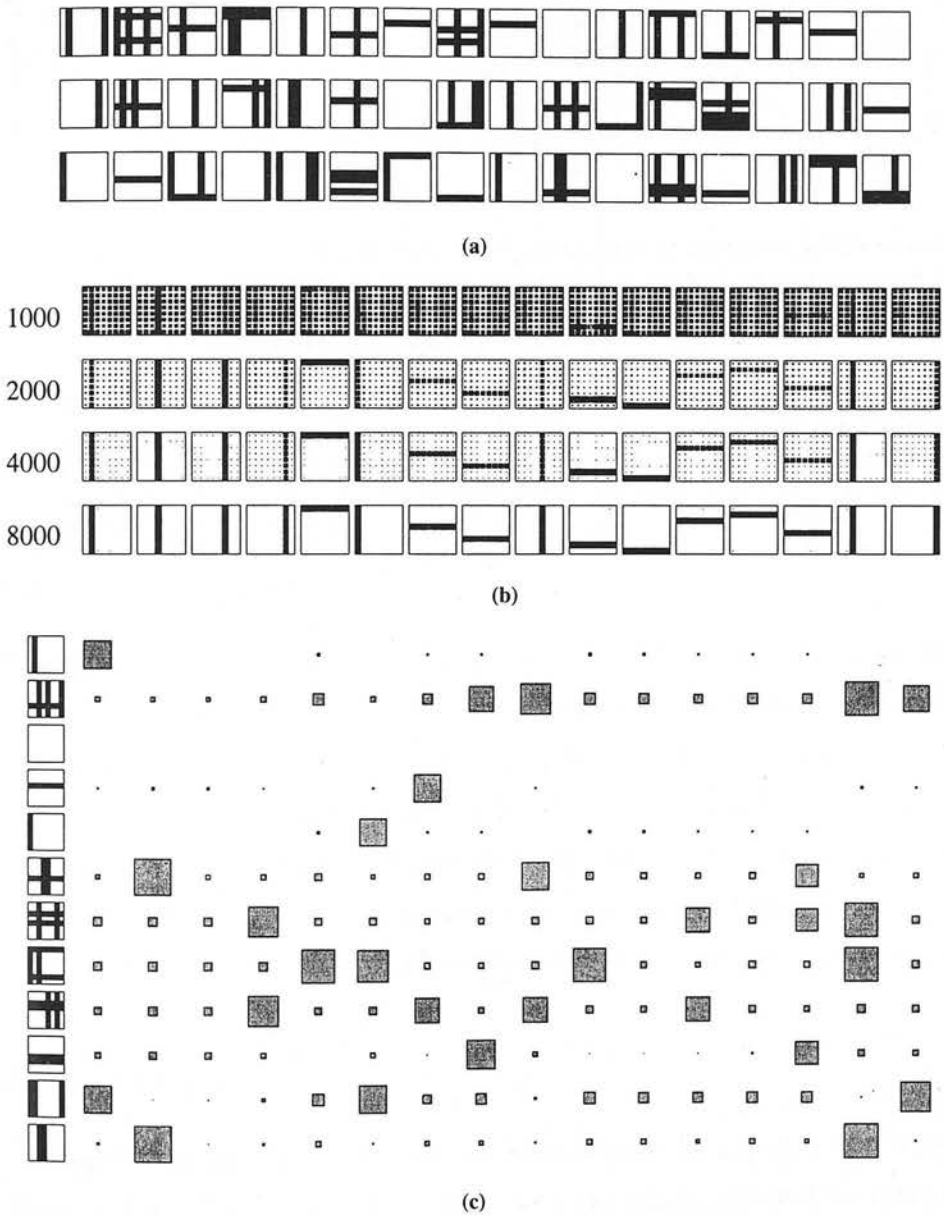


Figure 5.2: Learning a distributed code. (a) 48 typical input patterns for the bars problem. Bars on an 8x8 grid are active with probability $\frac{1}{8}$. Dark squares represent active inputs in the grid. (b) The synaptic weights connecting nodes to these grid points for the 16 nodes of a network after training on 1000, 2000, 4000 and 8000 input patterns. The size of each square is proportional to the magnitude of the synaptic weight scaled by the largest weight for all nodes at that time. (Weights are scaled by row so that the small weights at the start of training will be visible.) (c) The response of the network after training on 4000 input patterns. The left-most column shows the input pattern, with the remainder of each row showing the response of the network to this pattern. The size of the response of each node, in the same order as the RFs are shown in (b), is represented by the size of each square.

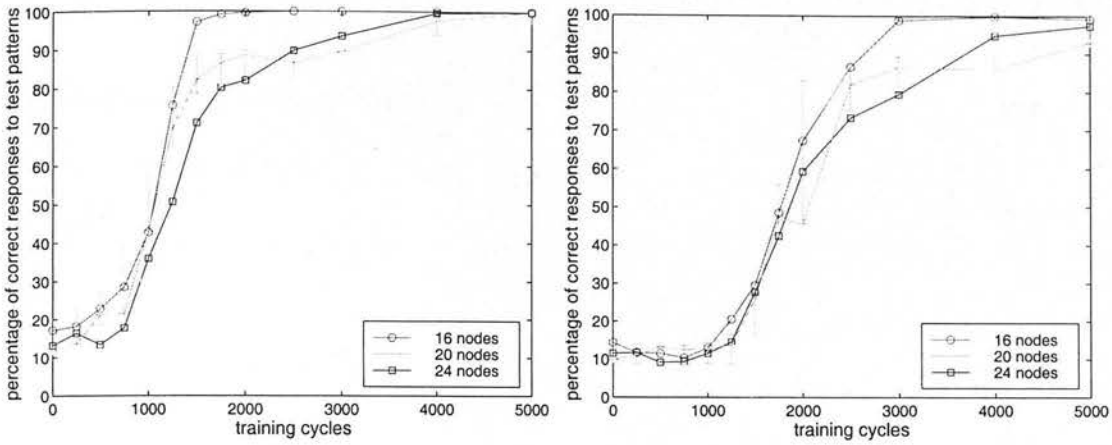


Figure 5.3: The change in the accuracy of the response of the network with training time. The percentage of complete input patterns that are correctly represented by the response of the network is shown. Values are calculated by assuming that nodes ought to respond to the preferred stimulus which is learnt by the end of training (*i.e.*, after 5000 iterations). The value at iteration I is calculated from the number of correct responses given for the next 500 training patterns after I. Results are shown for typical runs of networks containing 16, 20 and 24 nodes. (a) Accuracy calculated from the response of the n most active nodes for input data consisting of n bars. (b) Accuracy calculated from the response of all nodes with activation exceeding a predefined threshold. The error bars show the maximum and minimum accuracy values found for a 20-node network over the course of 20 successful experiments with random variations in the training order.

The improvement of the accuracy of the response of the network over time, measured using both these criteria is shown in figure 5.3. It can be seen that the accuracy when evaluated using the n most active nodes (figure 5.3(a)) improves more quickly than the accuracy evaluated by the response of nodes with activities exceeding a threshold (figure 5.3(b)). This is due to less *a priori* knowledge being applied in the latter case which relies entirely on there being significant differentiation between the activation values of active and inactive nodes in order for thresholding to be effective, whereas the former case is less sensitive to the relative output values of nodes which should be active and those that should not be.

5.1.1 Robustness

Although the learning algorithm encourages nodes to represent a single unique input there is no guarantee that this will happen (the results above are for a typical experiment in which a suitable encoding was found). The results were sensitive to variations in the training order (variations caused by different randomly generated training sets and different values for the noise used to select the winning node) such that with 16-node networks one node often came to respond to two separate bars (which were thus indistinguishable by the network) while another node represented none. Further training failed to improve the representation. To test the robustness of the algorithm to the training order a network consisting of 16 nodes was trained using 54 sets of randomly generated training data. Of these experiments only

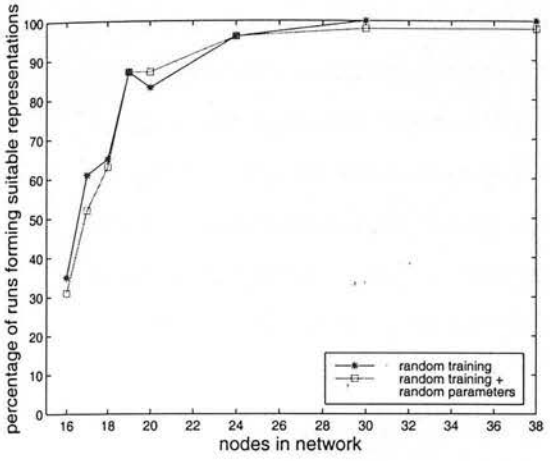


Figure 5.4: The change in robustness with number of nodes in the network. The percentage of experiments, of 4000 training cycles in length, for which the nodes form suitable weights to represent all the bars is shown. Results are given for experiments performed with variations in the training order and for randomly selected parameters of the learning algorithm as well as variations in the training order.

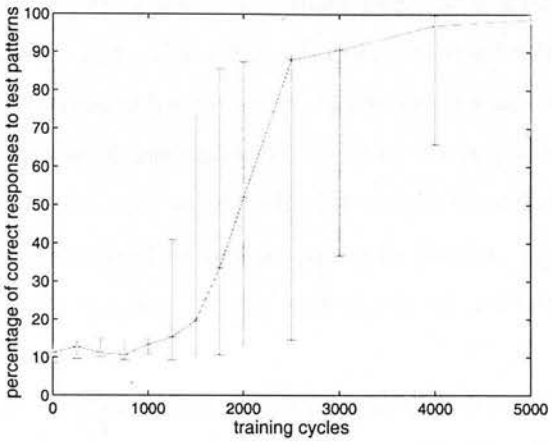


Figure 5.5: The robustness of the change in the accuracy of the response of the network with time. The percentage of complete input patterns that are correctly represented by the response of a 20-node network calculated as for figure 5.3(b). A typical result is shown for parameters randomly chosen in the range $\pm 50\%$ from the hand-picked values used above. The error bars show the maximum and minimum accuracy values found over the course of 20 successful experiments with random variations in the training order and in the parameters values.

35% learnt to use each node to represent a single bar. However, the robustness was improved by using more nodes than there were independent input patterns (reaching 100% for a network of 30 nodes or more, see figure 5.4). In cases where excess nodes were used, although the allocation of specific bars to specific nodes was still sensitive to changes in the training order, the algorithm robustly found a subset of 16 nodes, each of which represented a single unique bar.

The robustness of the network to changes in the parameters of the learning algorithm was also tested. Networks consisting of 16 nodes trained using 54 sets of random parameters (with values uniformly chosen in the range $\pm 50\%$ from the hand-picked values used above) learnt a suitable representation in only 31% of cases, while networks consisting of more nodes, tested in the same manner, had increasing robustness (see figure 5.4). These results are extremely similar to the effects of modifying the training order. Since randomising the network parameters also resulted in random modifications to the training order this suggests that the network is much less sensitive to changes in parameter values than to changes in the learning order. However, although modifying the parameters has little effect on the robustness of the algorithm in finding a suitable representation, it does affect the speed at which that representation is found. It can be seen from figure 5.5 that there was considerable variation in the accuracy of response over time recorded during these experiments with a 20 node network. Note that many

of these experiments were faster at learning an accurate representation than when using the hand-picked parameter values. Given that the above result shows that the algorithm is relatively insensitive to considerable variation in all the parameters simultaneously no attempt has been made to test the robustness of the algorithm to changes in individual parameter values (which would surely be at least as good). Such experiments would be extremely time consuming since sensitivity to variations in the training order is the dominant effect requiring that all experiments be repeated many times to produce meaningful comparisons. For the same reason in all experiments reported in this thesis typical results are shown, which have been generated using the standard parameter values. The fact that the algorithm has been successfully applied to many different problems without the need for parameter tweaking provides further evidence (in addition to the explicit test reported above) that the algorithm is reasonably insensitive to the parameter values.

The algorithm, using the standard parameters and the same input data, was tested (once) with networks varying in size from 16 to 33 nodes. All these networks, except that with 17 nodes, found a suitable representation (as did the 17-node network when trained using a different random training order). The algorithm is thus robust to changes in the number of nodes used in the network. A network consisting of 20 nodes (trained using the standard parameter values) found a suitable representation, in terms of weight decoding, within 1700 presentations (see figure 5.6). The accuracy of the encoding was tested using activity decoding with a further 500 patterns, after training for 4000 cycles. The response generated by this network (for input data consisting of n bars) was such that the n most active nodes were those which correctly represented all the active bars in 69 out of 84 (82.1% of) unseen test patterns (and 413 out of 416 previously seen test patterns). The response of individual nodes to the presence of individual bars within these test patterns was more reliable with the n most active nodes correctly responding to 315 out of 334 (94.3% of all) bars presented within the unseen test patterns. Both these figures rose to 100% after further training. Nodes which had an activity value exceeding a threshold correctly represented all the active bars in the input pattern (and only those bars) for 65 out of 84 (77.4% of) unseen test patterns (and 413 out of 416 previously seen test patterns). Unseen test patterns are those which consist of previously unencountered combinations of bars and hence are likely to be predominantly patterns containing a large number of active bars. The greater the number of active bars in a test pattern the less distinct active nodes become with respect to the activity level of other nodes. This is due to the overlap between perpendicular bars, so that an input pattern that contains several bars of one orientation will partially activate all the nodes representing bars at the perpendicular orientation. Testing using a randomly generated test set, that will include previously seen as well as unseen patterns, has less bias towards patterns containing a large number of bars and hence gives higher accuracy levels. Figure 5.3 shows how these accuracy values improve during training for three sizes of network. It can be seen that the profile of accuracy verses training time is very similar for all network sizes and remains

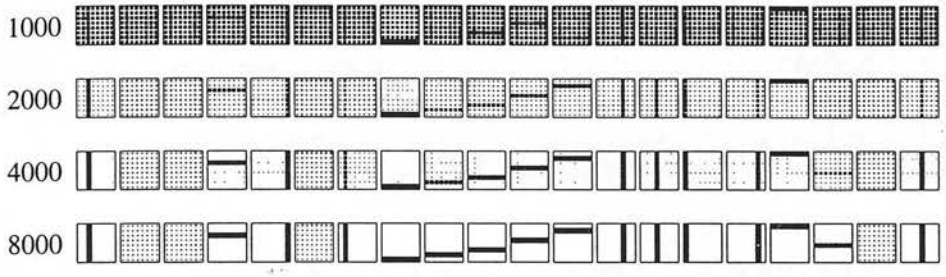


Figure 5.6: Learning a distributed code with excess nodes. The synaptic weights for the 20 nodes in a network after training on 1000, 2000, 4000 and 8000 input patterns. The size of each square is proportional to the magnitude of the synaptic weight scaled by the largest weight for that row.

similar when experiments are repeated with random variations to the training order (shown as error bars on the results of the 20-node network).

So far all the experimental results have been for perfect input patterns. The network can still learn the bars problem even when the input data is corrupted with a considerable level of independent noise. Random values uniformly selected from the range $-v$ to $+v$ were added to each input and the resulting values truncated to be in the expected range of 0 to 1 (figure 5.7(a) shows examples of such patterns for $v = 0.5$). In this case a network of 16 nodes, using the same learning parameters as above, failed to find a solution for most values of v. However, a more robust network of 20 nodes found a suitable coding (for $v = 0.3, 0.4, 0.5, 0.6$ and 0.7). The noise slows down learning so that a longer training period is required. With a network of 20 nodes, and $v = 0.5$, a suitable solution was found, in terms of weight decoding, after 3000 presentations (figure 5.7(b)). The response of individual nodes to the presence of particular bars in the input pattern is fairly clear given sufficient training (figure 5.7(c)). When tested with a further 500 patterns, after training with 4000 pattern presentations, the response generated by this network (for input data consisting of n bars) was such that the n most active nodes were those which correctly represented all the active bars in 486 out of 500 (97.2% of) unseen test patterns (in this case all patterns are unseen, not just ones containing previously unencountered combinations of bars, due to the noise). In general, noise slowed down learning so that the weights were less differentiated, than before, after a given number of training cycles. In addition, noise in the inputs generates noise in the outputs due to nodes which ought to be inactive being partially activated by inputs which are not parts of the bar patterns, and nodes which ought to be active being less activated by bars which contain reduced input values. Hence, the difference between an active and an inactive node is reduced by the noise and more training is needed to make weights sufficiently selective to overcome this reduction in discernibility. With perfect data it was noted that the overlap between patterns causes those nodes which are active to become less distinct as more bars are present in the input pattern. The same problem is seen with the response to noisy data but starts to occur with fewer active bars in the input data due to the additional, partial, activation of nodes by the background noise.

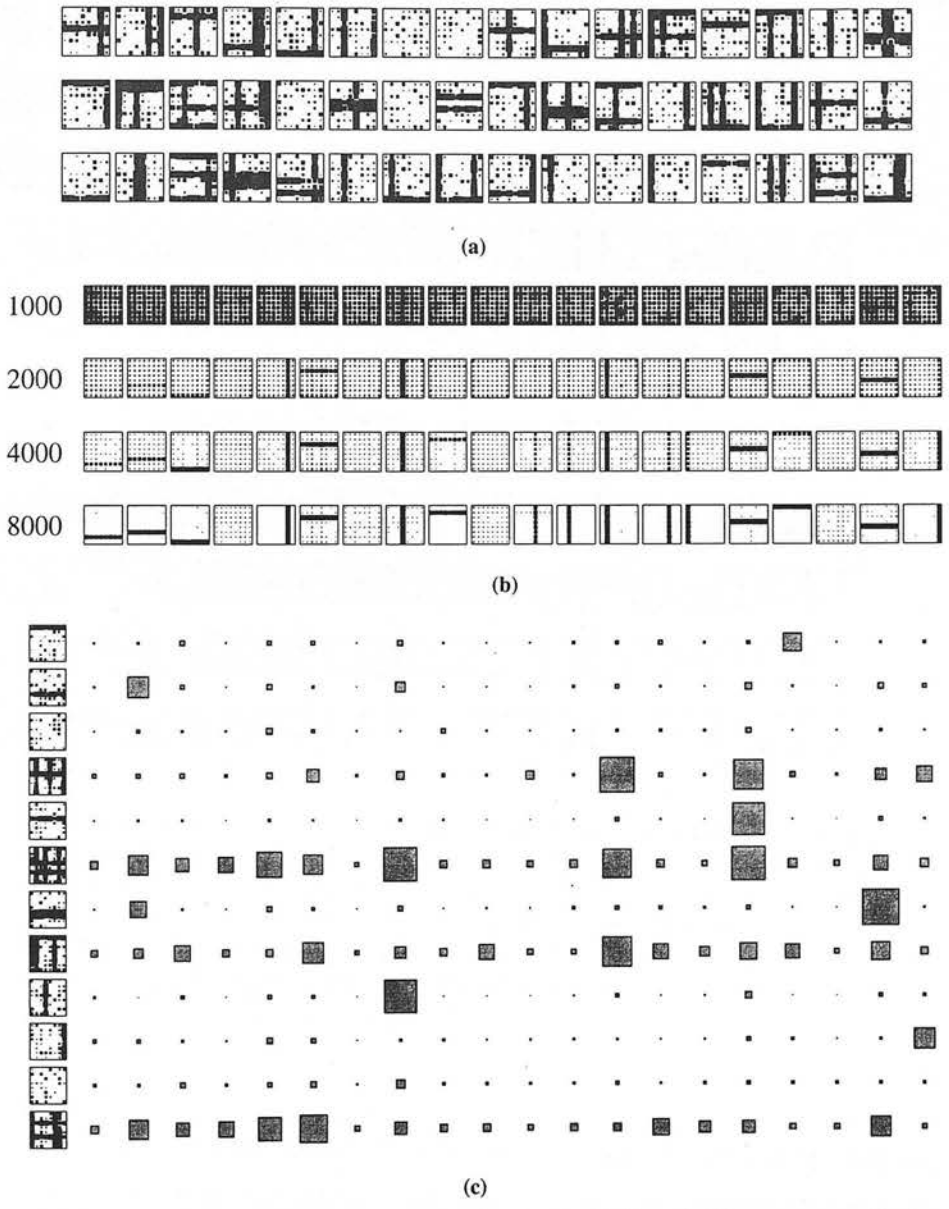


Figure 5.7: Learning a distributed code with noisy data. (a) 48 typical input patterns for the noisy bars problem. The size of each square is proportional to the activation of that input in an 8x8 grid. (b) The synaptic weights for the 20 nodes in a network after training on 1000, 2000, 4000 and 8000 input patterns. The size of each square is proportional to the magnitude of the synaptic weight scaled by the largest weight for that row. (c) The response of the network after training on 4000 input patterns. The left-most column shows the input pattern, with the remainder of each row showing the response of the network to this pattern. The size of the response of each node, in the same order as the RFs are shown in (b), is represented by the size of each square.

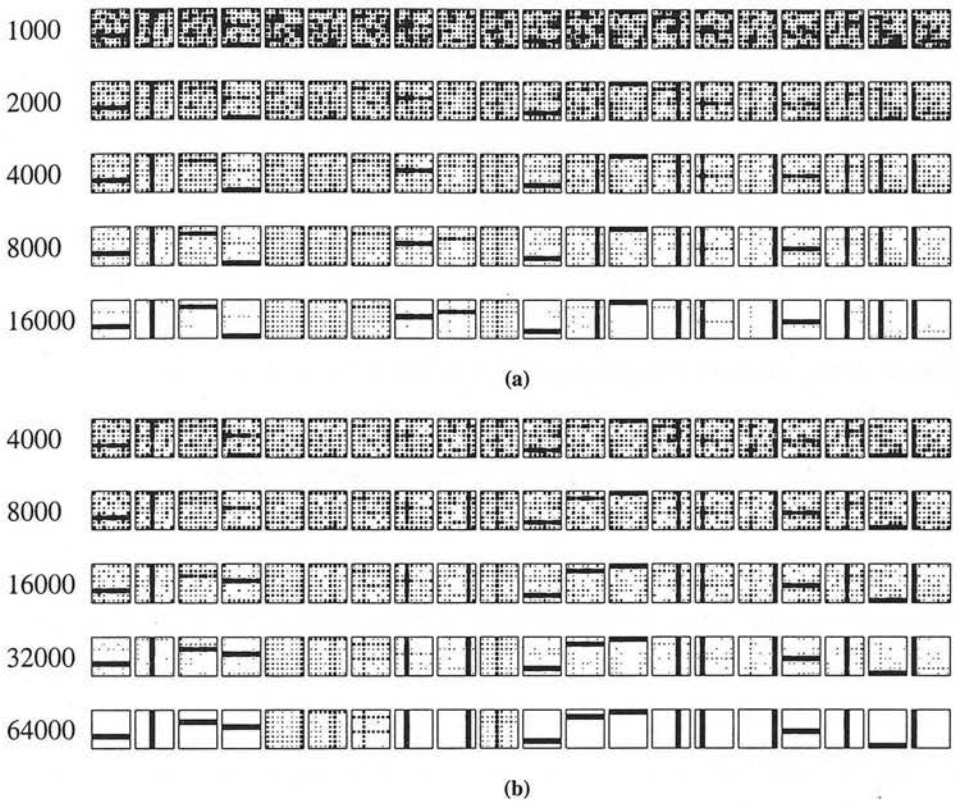


Figure 5.8: Learning a distributed code with randomly initialised synaptic weights. (a) The synaptic weights for the 20 nodes in a network after training on 1000, 2000, 4000, 8000 and 16000 input patterns. The weights were initialised to be random values uniformly selected from the range 0 to 0.01. (b) The synaptic weights for the 20 nodes in a network after training on 4000, 8000, 16000, 32000 and 64000 input patterns. The weights were initialised to be random values uniformly selected from the range 0 to 0.1. The size of each square is proportional to the magnitude of the synaptic weight scaled by the largest weight for that row.

The network is also robust to noise in the initial values for the synaptic weights. Initialising the synaptic weights to be random values uniformly selected from the range 0 to v did not prevent a 20-node network from finding a suitable representation of the bars problem for any value of v that was tested. However, the speed with which a solution was found was reduced, requiring more iterations as v increases, until approximately 12000 training cycles are required to find the solution for all large values of v (large values of v cause node activations exceeding 1, resulting in weights being scaled downwards, so that results are very similar for all values of v greater than 0.05). Figure 5.8 shows the development of weights for a 20-node network initialised with random weights. Note that using random values for the weights not only gives random receptive fields but also random lateral weights.

The network is robust to node failure. If a node representing one of the bars in a 20-node network was removed then one of the nodes which was not previously representing a bar quickly became responsive to that bar (within 1000 cycles) and subsequently took over representing that input pattern (figure 5.9).

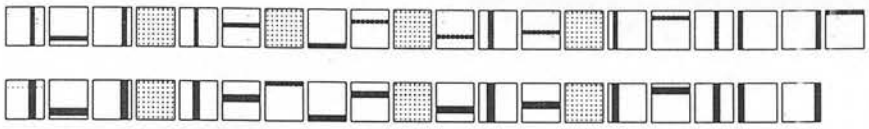


Figure 5.9: Robustness to node failure. The top row shows the synaptic weights for a 20-node network after training for 4000 cycles. A node was removed from this network and training continued. The bottom row shows the synaptic weights after a further 4000 training cycles. It can be seen that the seventh node from the left, that was not representing any inputs after the initial training, has come to represent the preferred input of the node which was removed.

5.1.2 Stability

For any network containing more than 16 nodes there are nodes which do not represent any input pattern. As can be seen from figures 5.6 and 5.3 the representation formed is stable in such conditions, with the unused nodes remaining unresponsive to all inputs[3]. (The representation formed when there are too few nodes, with some nodes representing multiple bars, is also stable.) It is thus not necessary to know beforehand how many patterns are present in the data. Hence, a network with excess nodes can be used to ensure that an appropriate representation is formed. Such a network has excess variance (section 2.3.3.1), but in this form the variance does not make finding an appropriate solution more difficult. Nodes which do not represent an input pattern will still be selected as the inhibiting node (due to habituation keeping all nodes winning the competition with approximately equal frequency). However, the weak weights of unused nodes are insufficient to block the inputs to other nodes, and hence, those nodes which better represent the current inputs are still activated, and update their synaptic weights more strongly than the inhibiting node. The synaptic weights of unused nodes thus remain small and undifferentiated. This can be seen by examining the development of the weights in a 20-node network. Figure 5.10 shows that synaptic weights become strongly differentiated in nodes which do represent an input pattern, while the synapses of nodes which do not represent a pattern all have very similar weights which are of intermediate strength.

It would be simple matter to identify unused nodes: they have small, undifferentiated, weights and are never the most active node (even when chosen as the winning node). For networks which have random initial synaptic weights, and to a lesser extent those trained with noisy data, unused nodes have more variation in the strength of their weights. However, they can still be easily identified since the weights are still small and relatively undifferentiated compared to other nodes. Having identified unused nodes they could be pruned from the network to leave only those nodes that do represent the data. Conversely, new nodes could be added to the network if required. The addition of new nodes (with all synaptic weights initialised to zero) will have little effect on the encoding if all input features are already

[3] An algorithm that used the standard form of lateral inhibition would be unstable in this case (see section 4.3.2.1) and so would require *a priori* information about the number of features in the input space.

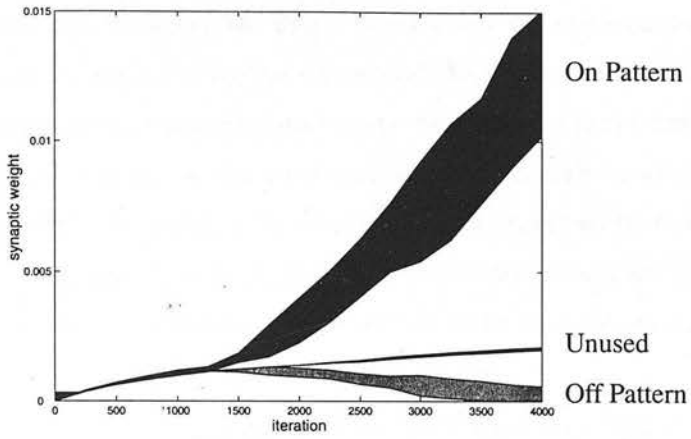


Figure 5.10: Change in strength of synaptic weights with time during learning the bars problem. The range of synaptic weight values is shown for a 20-node network. Weights for nodes which represent bars become differentiated with strong weights forming to sources that correspond to the preferred input pattern (labelled 'On Pattern') and weights to other grid points being reduced towards zero (labelled 'Off Pattern'). The synaptic weights, to all inputs, for nodes which do not represent a bar (labelled 'Unused') grow very slowly and are all very similar.

represented by the original nodes: the new nodes will fail to compete with the original nodes and will retain their small, undifferentiated, weights for the same reasons that unused nodes remain unused. This will be true even if the input features are not suitably represented (*e.g.*, if a 15-node network was applied to the bars problem and one node came to represent two bars, this node would prevent any new node from representing either of these bars). In order for additional nodes to be able to compete with existing nodes it is necessary for them to have their weights initialised in order to do so. It is generally the case, for any constructive neural network, that nodes are added which have predefined weights. The weights of the new node can be initialised to be 1) similar to the weights of an existing node or have weights with values intermediate between those of more than one existing node, or 2) representative of an input pattern. The criteria for adding a new node could be that an input pattern is not well represented (*i.e.*, all the node activations are low on presentation of that input), or that an existing node is responsive to too many input patterns. It can be concluded that it would be possible to modify the algorithm to add and delete nodes to find the optimum network topology to represent input distributions. However, the current implementation does not do this.

The algorithm is able to form a stable representation of input distributions in which subfeatures do not appear together with equal frequency. An example of such an input distribution is the encoding of the shape and colour of an object (figure 4.3 section 4.1.2). Different colours are mutually exclusive events (as are different shapes) and should never occur simultaneously, although all combinations of shape and colour should be able to be represented together[4]. The bars data was modified to generate

[4] An algorithm that used the standard form of lateral inhibition would be unable to form a stable representation of this data (see section 4.3.2.1) and would generate spurious outputs.

an input distribution of this type by allowing, at most, a single horizontal and a single vertical bar to occur together: a single horizontal bar, at a random position, occurred with probability 0.75 and a single vertical bar, at a random position, occurred with probability 0.75 (figure 5.11(a)). When a network of 16 nodes was trained on this data it found a suitable representation, where each node represented exactly one bar and each bar was represented exactly once. As before, during training, the weights corresponding to the represented bar become much stronger than connections to inputs from other grid points (figure 5.11(b)), so that each node was most strongly connected to inputs from its preferred pattern within 2000 training cycles. The response of nodes thus became specific to the presence of that bar in the input data (figure 5.11(c)). After 4000 iterations, nodes which had an activity value exceeding a threshold correctly represented the active bars in the input pattern (and only those bars) for 100% of test patterns. No spurious activations were generated.

A local encoding is appropriate when the input data consists of mutually exclusive events. To demonstrate that the same algorithm can represent mutually exclusive input patterns the bars data was modified so that only one bar was active for any one presentation[5]. When a network of 16 nodes was trained on this data it found a suitable representation, in terms of weight decoding, within 2600 presentations. Although this problem would appear easier to solve, since each pattern is presented in isolation, it takes longer to learn due to there being fewer examples of each bar contained in the training data. (Previously bars have appeared with independent probability of $\frac{1}{8}$ so that the average number of bars in each pattern was 2, while in this case each pattern contains a single bar.) As before, during training, the weights corresponding to the represented bar become much stronger than connections to inputs from other grid points (figure 5.12(a)) and the response of nodes becomes specific to the presence of that bar in the input data (figure 5.12(b)). Moreover, when this network, having been trained using single bars only, was tested with patterns containing multiple bars it generated a correct distributed code with all nodes corresponding to bars in the input being active simultaneously (figure 5.12(c)). When tested with a further 500 patterns, after training with 4000 pattern presentations, the response generated by this network (for input data consisting of n bars) was such that the n most active nodes were those which correctly represented all the active bars in 475 out of 500 (95.0% of) unseen test patterns. Hence, the network provides generalisation by using multiple active nodes to represent novel input patterns.

Although the representation generated by the network should be stable for a given input distribution it should be plastic in the face of changes to that input distribution. To illustrate, consider a problem in which a network is initially trained exclusively with bars of one orientation, followed by bars of the opposite orientation. The bars data was modified to input only horizontal bars for I_1 iterations followed

[5] A suitable coding for this input distribution could be found using the standard form of lateral inhibition (see section 4.3.2.1), however, a completely different algorithm would be required from that which could be used to represent multiple, co-active, bars.

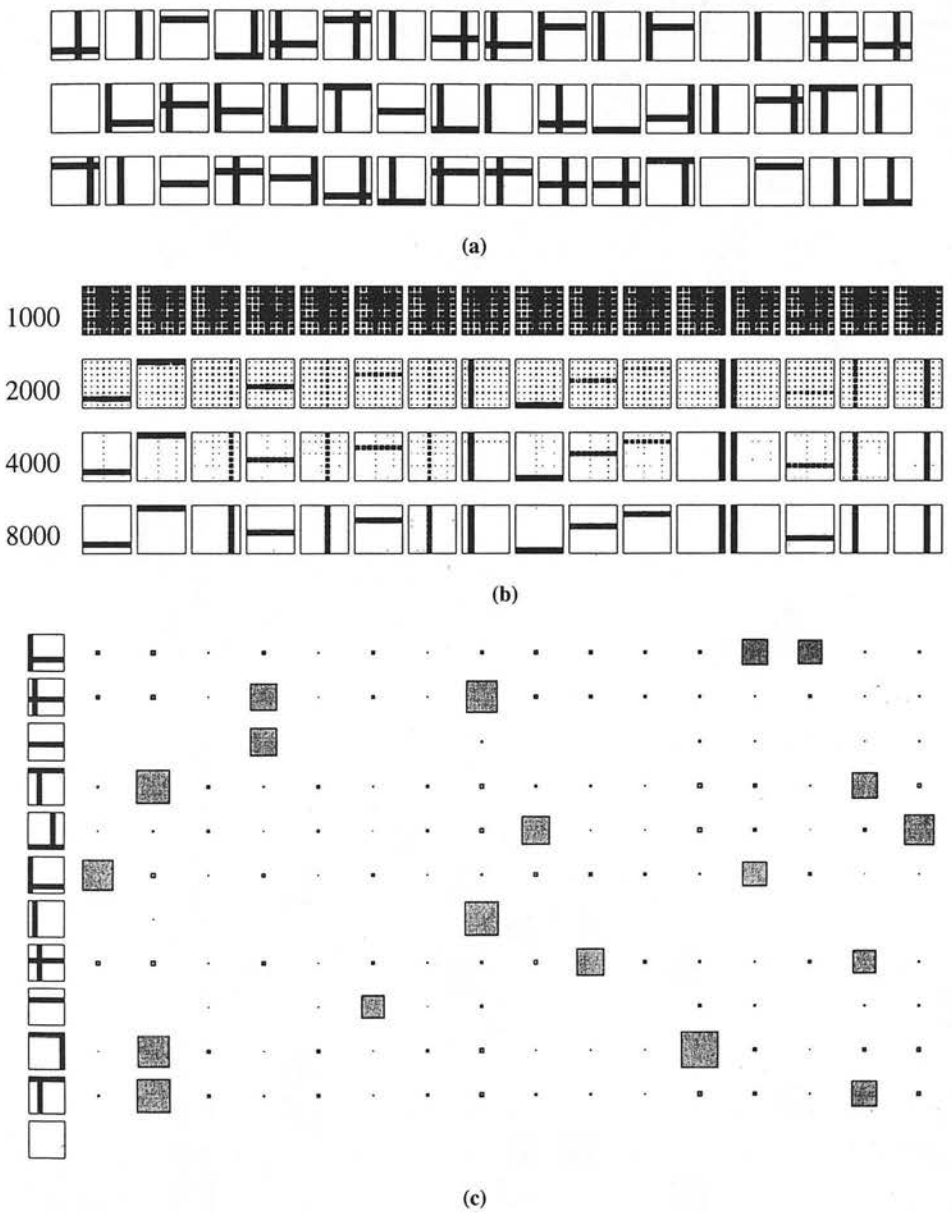


Figure 5.11: Learning a distributed code with some mutually exclusive subfeatures. (a) 48 typical input patterns for a modified bars problem in which neither multiple horizontal nor multiple vertical bars occur simultaneously. The size of each square is proportional to the activation of that input in an 8x8 grid. (b) The synaptic weights for the 16 nodes in a network after training on 1000, 2000, 4000 and 8000 input patterns. The size of each square is proportional to the magnitude of the synaptic weight scaled by the largest weight for that row. (c) The response of the network after training on 4000 input patterns. The left-most column shows the input pattern, with the remainder of each row showing the response of the network to this pattern. The size of the response of each node, in the same order as the RFs are shown in (b), is represented by the size of each square.

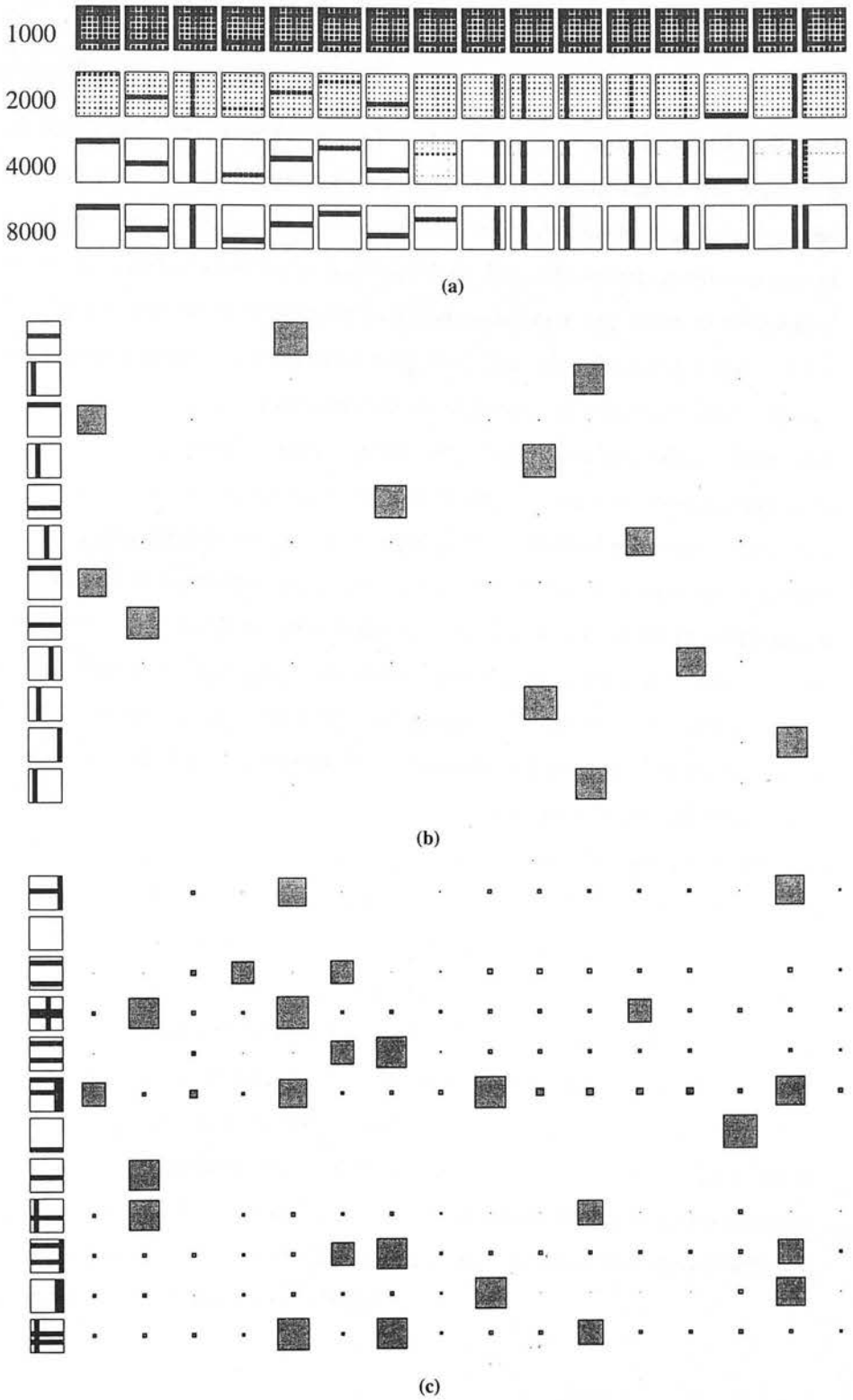


Figure 5.12: Learning to represent mutually exclusive input patterns. (a) The synaptic weights for the 16 nodes in a network after training on 1000, 2000, 4000 and 8000 input patterns. The size of each square is proportional to the magnitude of the synaptic weight scaled by the largest weight for that row. (b) The response of the network after training on 4000 input patterns. The left-most column shows the input pattern, with the remainder of each row showing the response of the network to this pattern. The size of the response of each node, in the same order as the RFs are shown in (a), is represented by the size of each square. (c) The response of the network to input patterns containing multiple bars after training on 4000 input patterns containing single bars.

by only vertical bars for I_2 iterations. Using a network of 8 nodes an encoding in which each node represented one horizontal bar was initially formed. With subsequent training, on vertical bars, each node came to represent one vertical bar. Since each new pattern had the same degree of overlap with each old pattern, the initial weights, selective to horizontal bars, had little effect on the learning of preferences to vertical bars, so that the time taken for each node to become responsive to one vertical bar was approximately the same as the time taken for learning the initial code. (However, in order for the weights to the new preferred input to become stronger than the weights to all other grid points required that I_2 be just longer than I_1.) For a network with 16 nodes, applied to the same problem, there were many unused nodes after the initial training phase. Many of these unused nodes (6) subsequently came to represent the vertical bars leaving most of the representations of horizontal bars intact. However, a few nodes (2) which had come to represent a horizontal bar subsequently came to represent a vertical bar. For networks with even more nodes it was still the case that the new data reused a few nodes that were representing old data rather than using only unused nodes. A less significant change in input distribution is provided by adding the new data to the initial input distribution, rather than replacing it. Horizontal bars only were used to train a network for I_1 iterations followed by both horizontal and vertical bars for I_2 iterations. A network of 16 nodes, using the standard parameters and tested with a single training sequence, was unable to find a suitable representation, for all the bars, for values of I_1 exceeding 1100. A more robust network of 20 nodes was able to learn to find a suitable representation of all horizontal and vertical bars for values of I_1 up to 4300. It thus seems that after a critical point in training with bars of one orientation the network becomes less able to represent new patterns. A node representing a horizontal bar can become relatively strongly activated by a vertical bar due to the overlap in the patterns. Such a node will then prevent unused nodes from successfully competing for the right to represent that vertical bar. In contrast, when the original patterns are entirely replaced with a new input distribution there is no problem for a node originally representing a horizontal bar to come to represent a vertical bar instead. Hence, after a critical period, the network becomes less plastic to modifications to the input distribution, but remains plastic in the face of significant changes to the input distribution (*cf.*, critical periods in neocortical development; section 3.2.3).

This section has shown that an identical algorithm can learn a local or a distributed representation as appropriate to the statistics of the input data. These representations are stable given a fixed input distribution but plastic to (certain) changes to the input distribution. The algorithm has been shown to be robust in the face of randomness in the parameter values, noise in the input values, randomness in the initial synaptic weight values and to the number of nodes in the network.

5.2 Representing Ambiguous Patterns

This section considers some very simple input distributions. These problems are easy to understand while illustrating some theoretical properties required in order to learn useful representations (Marshall, 1995a; Marshall and Gupta, 1998; Nigrin, 1993).

5.2.1 Overlap

Consider the simple problem shown in figure 5.13(a). There are three input sources, which for simplicity are labelled a, b and c. Two input patterns are presented to the network; 'ab' and 'abc'. In order to successfully represent this input data feedforward weights such as those shown in figure 5.13(b) must form, so that the two nodes learn to respond to one each of the two patterns. The nodes are labelled 'AB' and 'ABC' for ease of reference. Because these input patterns overlap, when the input is 'ab' the node representing 'abc' is also partially activated, while when the input is 'abc' the node representing 'ab' is fully activated. The activation rules ought to ensure that the output of the network represents large, complete, patterns in preference to small ones or incomplete ones (Marshall, 1995a; Marshall and Gupta, 1998). If the input sources represented the letters of words then the problem is one of distinguishing the word "cat" from the word "at". A more complex example (requiring a larger network) would be to distinguish the words "my", "myself", "self", and "elf" (Marshall, 1995a; Cohen and Grossberg, 1986).

If all strong feedforward weights are equal, perhaps due to saturation of the weights at an upper bound, then when the input is 'abc' node 'ABC' receives the most activation. Node 'ABC' can then suppress the activity of the other node, via lateral inhibition, to correctly represent the input. However, when the input is 'ab' both nodes receive the same excitation. The winner of the competition will be decided randomly, and hence, there is no guarantee that the correct representation will be generated (see table 5.1). Solutions to this problem have suggested modifying the excitation received by each node, using a Webber Law rule (Marshall, 1995a), such that:

$$y = \frac{\zeta_1 \sum qx}{\zeta_2 + \sum q}$$

(where ζ_1 and ζ_2 are constants); or using a masking field (Cohen and Grossberg, 1987; Marshall, 1995a) such that:

$$y = \frac{\sum qx}{\sum q} + \mathtt{fn}(y) \sum q.$$

These solutions scale very badly with the number of overlapping units (*i.e.*, for two nodes, one representing n inputs and the other representing $n+1$ inputs, such that there are n common inputs, the difference between activation values reduces with increasing n). In addition, it is more common for neural networks to normalise the sum of the synaptic weights, rather than having all connection strengths equal. In the case of pre-synaptic normalisation, the same ambiguity exists in the response to input 'ab'. In the

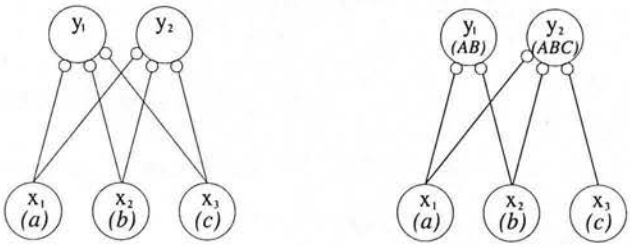


Figure 5.13: A neural network architecture for learning to represent two simple, overlapping, patterns. (a) The network consists of two nodes, each of which receives three inputs; a, b, and c. (b) Only highly weighted connections are shown which are required in order for the nodes to represent input patterns 'ab' and 'abc'. In all subsequent figures in this section which show network architectures only the highly weighted connections necessary for the task will be shown although all networks are fully connected.

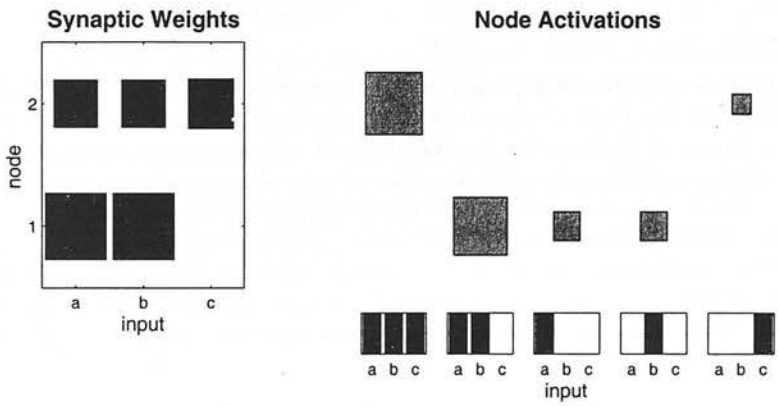


Figure 5.14: Learning overlapping input patterns. Two nodes are trained using input patterns 'ab' and 'abc'. Results are shown after 4000 training cycles. The strength of synaptic weights between each node and each input are shown on the left using squares which have sides of length proportional to the synaptic weight. Various input patterns are illustrated along the bottom of the right side, active inputs being represented by dark squares. The response of each node to these input patterns, averaged over a number of trials, is shown on the right using squares which have sides of length proportional to the node activations. Each node learns to represent one of the input patterns. The synaptic weights are such that each node responds to its preferred pattern and makes no response to the other training pattern, despite the overlap between the two patterns. Partial input patterns cause a weaker response from that node which has the closest matching preferred input.

case of post-synaptic normalisation it is the input 'abc' that is indistinguishable (see table 5.1). Neither the Webber Law nor the masking field resolves the ambiguity when normalisation is used.

The problem of distinguishing these overlapping patterns is really due to placing constraints on the allowable synaptic weights which serve to prevent the input patterns being divided in weight space. It is easy to see that there are sets of weights which unambiguously classify the two overlapping patterns (an example is given in table 5.1). Allowing overlapping input patterns to be distinguished by learning appropriate weights is not only more elegant but it scales better with the number of overlapping inputs. It would be possible to have different weights to each input depending on its saliency for recognising that pattern, so that inputs which more unambiguously indicated one pattern could be given higher weight.

Weight Distribution	Node	Input Pattern 'ab'	'abc'
Equal Weights	'AB'	$\overbrace{q+q=2q}$	$q+q=2q$
	'ABC'	$\underbrace{q+q=2q}_{ambiguous}$	$q+q+q=3q$
Normalised Weights ($q'=\sum q$)			
Post-synaptic Normalisation	'AB'	$\frac{q'}{2}+\frac{q'}{2}=q'$	$\overbrace{\frac{q'}{2}+\frac{q'}{2}=q'}$
	'ABC'	$\frac{q'}{3}+\frac{q'}{3}=\frac{2}{3}q'$	$\underbrace{\frac{q'}{3}+\frac{q'}{3}+\frac{q'}{3}=q'}_{ambiguous}$
Pre-synaptic Normalisation	'AB'	$\overbrace{\frac{q'}{2}+\frac{q'}{2}=q'}$	$\frac{q'}{2}+\frac{q'}{2}=q'$
	'ABC'	$\underbrace{\frac{q'}{2}+\frac{q'}{2}=q'}_{ambiguous}$	$\frac{q'}{2}+\frac{q'}{2}+q'=2q'$
Example Sensible Weights	'AB'	$0.5+0.5=1.0$	$0.5+0.5=1.0$
	'ABC'	$0.4+0.4=0.8$	$0.4+0.4+0.4=1.2$

Table 5.1: The excitation received by two nodes representing two simple, overlapping, patterns. The activation received by the two nodes shown in figure 5.13 for each input pattern under different conditions for the weight strengths. For the 'equal weights' condition all connection strengths are the same (equal to q). In this case both nodes respond equally strongly to the input pattern 'ab'. For the 'normalised weights' condition the sum of connection strengths to each node, or from each source, is the same (equal to q'). For post-synaptic normalisation both nodes respond equally strongly to the input pattern 'abc', while for pre-synaptic normalisation both nodes respond equally strongly to the input pattern 'ab'. In the 'example sensible weights' condition all weights to node 'AB' are 0.5, and all weights to node 'ABC' are 0.4. In this case each node successfully responds most strongly to its preferred input pattern.

Hence, if patterns had a significant overlap the few inputs which were not shared would be more salient for recognising those patterns and could be given a higher weighting.

Since the learning algorithm developed for this thesis does not normalise weights or constrain them in any other way, except through the learning rules, it generates a solution to this problem by finding suitable weights to distinguish between the patterns based on the excitation received[6]. Figure 5.14 shows the weights which are learnt and the resulting activation of each node on presentation of these patterns. The total synaptic weight to the node representing 'abc' is slightly higher than that to the node representing 'ab'. Thus when the input is 'abc' and both nodes receive maximum input the node

[6] In order to prevent unconstrained weight increases a crude method has been used of normalising the weights, once a maximum activation value has been reached (described by equation 4.7). Once this condition is reached the network will have normalised weights and suffer the problems discussed above for this condition; causing equal activation of both nodes given an input of 'abc'. This does not prove to be a problem due to habituation preventing node 'AB' responding to both patterns. In addition, a dynamical implementation of the algorithm (*i.e.*, equation 4.13), in which the output is calculated after iteration, would behave correctly since node 'ABC' would receive uninhibited input from c, while 'AB' would have all of its input inhibited.

representing the larger pattern wins the competition. However, the synaptic weights to inputs a and b are smaller for the node representing 'abc' than for the other node, hence, the node representing 'ab' wins when the input is 'ab'. This network thus responds to the largest, complete, pattern. Such a property is important in order to correctly categorise inputs, and has been termed scale sensitivity (Marshall, 1995a): "large, complete, output representations are better than small ones or incomplete ones" (Marshall and Gupta, 1998).

When novel, partial, patterns are presented to the network the best matching node is activated in isolation, such that the strength of activation is proportional to the degree of overlap between the partial pattern and the preferred input of that node (figure 5.14). Hence, if the input is 'a' or 'b', which partially matches both of the training patterns, then the node representing the smallest pattern responds since these partial patterns are more similar to 'ab' than to 'abc'. When the input is 'c' this partially matches only one of the training patterns and hence node 'ABC' responds. The responses are weaker than the response caused by complete patterns so that the strength of activation is proportional to the degree of overlap between the partial pattern and the preferred input pattern. The next example illustrates how the algorithm deals with ambiguous partial input patterns.

5.2.2 Uncertainty

Consider the same neural network architecture as above applied to a different set of training data. Two input patterns are presented to the network; 'ab' and 'bc'. To successfully represent this input data we would expect strong feedforward weights to form as shown in figure 5.15, so that the two output nodes learn to respond to one each of the two patterns. In this case there is no difficulty in finding synaptic weights that correctly classify the input patterns. The weights that were learnt by the algorithm developed in this thesis are shown in figure 5.16. Each node responses strongly to its preferred stimulus and makes no response to the other stimulus. When tested with additional, novel, patterns (*i.e.*, 'a', 'b', 'c', 'ac', and 'abc') the behaviour of the network to partial patterns can be tested. It can be seen that each node responds at about half the maximum activation value when the input matches half its preferred input. Since the network can represent multiple input patterns occurring simultaneously this is also true when half of each pattern is presented (*i.e.*, 'ac'). The network thus finds the best match to a partial pattern and responds in proportion to how well that input matches the preferred input.

However, there is no unambiguous best match when the partial pattern equally activates both nodes. Input 'b' matches half of the preferred input patterns of both nodes, hence on presentation of 'b' each node receives equal activation. Competition between the nodes means that one will be selected as the winner and will respond to this input (at about half the maximum activation value) and will fully inhibit the other node. Since both nodes have an equally strong connection to input 'b' each will win the competition about half the time. The response of each node averaged over a number of trials is ap-

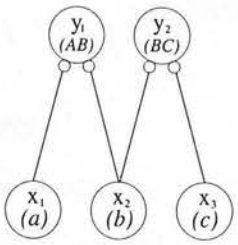


Figure 5.15: A neural network architecture for learning to represent two simple, overlapping, patterns. The network consists of two nodes, each of which receives three inputs; a, b, and c. Only highly weighted connections are shown which are required in order for the nodes to represent input patterns 'ab' and 'bc'.

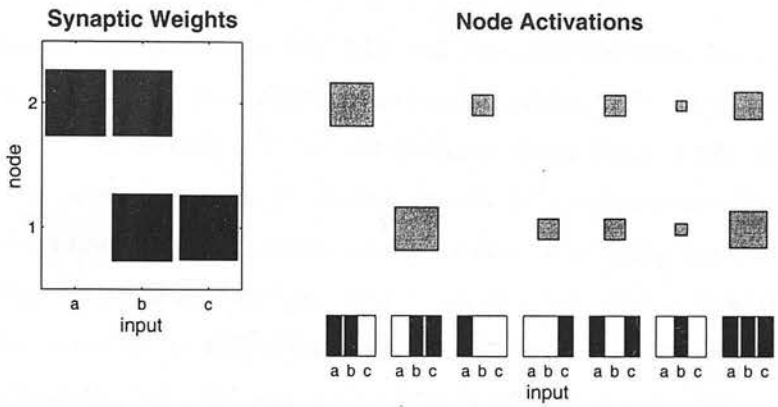


Figure 5.16: Learning overlapping input patterns. Two nodes are trained using input patterns 'ab' and 'bc'. Results are shown after 4000 training cycles. The strength of synaptic weights between each node and each input are shown on the left using squares which have sides of length proportional to the synaptic weight. The particular preferred input learnt by each node is arbitrary and hence the ordering of the nodes does not correspond to that shown in figure 5.15. Various input patterns are illustrated along the bottom of the right side, active inputs being represented by dark squares. The average response to these input patterns of each node over a number of trials is shown on the right using squares which have sides of length proportional to the node activations. Each node learns to represent one of these patterns. The synaptic weights are such that each node responds to its preferred pattern and makes no response to the other training pattern. Novel input patterns ('a', 'c', and 'ac') which each match half of a training pattern cause half the activation in the best matching nodes. A partial pattern ('b') which has two best matching nodes causes an average response of a quarter of the maximum value in both nodes. A pattern ('abc') which fully matches both preferred inputs causes an average response of three-quarters of the maximum value due to the nodes having preferred inputs which have half their subfeatures in common.

proximately one-quarter of the maximum response. When the input pattern is 'abc' both nodes receive maximum input. Each node will win the competition half the time. The winning node will inhibit input from 'b' reaching the other node but does not inhibit the other input. The losing node thus responds at half its maximum activation value. Over time the average response of each node is three-quarters of its maximum value. These last two input patterns illustrate cases where there are two equally good matches to an input. "When there is more than one best match for an input pattern, the best-matching representations [should] divide the input signals equally" (Marshall and Gupta, 1998). The algorithm succeeds in this only if the average response is taken since the selection of a single winning node at

each iteration causes a choice to be made between equally likely representations. A dynamical implementation of the algorithm (*i.e.*, equation 4.13), in which the output is calculated after iteration, could generate the correct responses for single presentations.

5.2.3 Exclusive Allocation

When six nodes are trained with input patterns 'a', 'ab', 'abc', 'cd', 'de', and 'def' synaptic weights are learnt which allow each node to respond to its preferred input pattern (figure 5.17). In the same way as above this network will respond correctly to partial and ambiguous inputs (figure 5.18). However, it fails to respond correctly to an input consisting of patterns 'ab' and 'cd' simultaneously. This is due to this combination of inputs causing the strongest activation of the node representing 'abc'. When the node representing 'abc' wins the competition for input 'abcd' it will inhibit all input reaching the node representing 'ab'. However, it will fail to prevent nodes with connections to input 'd' from all being activated and so nodes representing 'cd', 'de', and 'def' will all be partially active. Hence, there is a problem that an input which consists of two sub-patterns may more strongly activate a node which represents a pattern which overlaps with both of the sub-patterns. On occasions habituation allows the node representing 'ab' to win causing it to inhibit all input reaching the node representing 'a' but it causes no inhibition to inputs from 'c' or 'd', hence although node 'CD' will be, correctly, fully active all other nodes will be partially active. Similarly, when habituation allows node 'CD' to win the competition node 'AB' will be (correctly) fully active, but node 'A' and node 'ABC' will both be partially active. Hence, even when one of the correct nodes is selected as the winner there is still a problem. The result is that on average all nodes are partially active. The network thus fails to fully enforce the requirements of 'exclusive allocation' (Marshall and Gupta, 1998). This requires that all inputs are represented and that each input is only represented once. In the previous sections there was exclusive allocation since the ambiguous parts of the input pattern were fully represented by the winning node and a single other node. In the current example the part of the input pattern that is not the preferred input of the winning node is still ambiguous. However, there is no further competition for the right to respond to this part of the input pattern since the winning node is the only one that is allowed to inhibit others. Hence exclusive allocation fails: whichever node is selected as the winner it will not prevent other inputs from being represented by more than one output. This problem should again be solved by using a dynamical implementation of the algorithm since it would then not only be the inputs to the winning node that were exclusively allocated but also any other inputs.

The above examples have shown that a dynamical implementation of the algorithm would have several advantages over one in which the winning node is selected based on the feedforward activation only. A

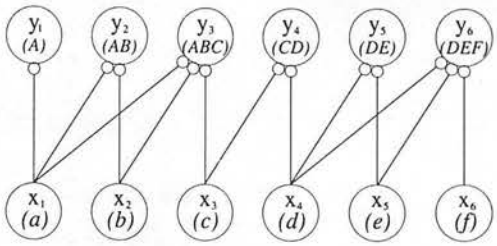


Figure 5.17: A neural network architecture for learning to represent multiple, overlapping, patterns. The network consists of six nodes, each of which receives six inputs; a, b, c, d, e, and f. Only highly weighted connections are shown which are required in order for the nodes to represent input patterns 'a', 'ab', 'abc', 'cd', 'de', and 'def'.

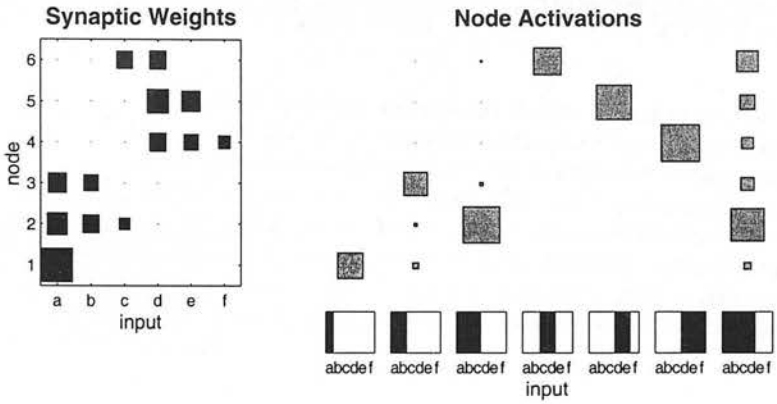


Figure 5.18: Learning overlapping input patterns. Six nodes are trained using input patterns 'a', 'ab', 'abc', 'cd', 'de', and 'def'. Results are shown after 4000 training cycles. The strength of synaptic weights between each node and each input are shown on the left using squares which have sides of length proportional to the synaptic weight. The particular preferred input learnt by each node is arbitrary and hence the ordering of the nodes does not correspond to that shown in figure 5.17. Various input patterns are illustrated along the bottom of the right side, active inputs being represented by dark squares. The average response to these input patterns of each node over a number of trials is shown on the right using squares which have sides of length proportional to the node activations. Each node learns to represent one of the input patterns. The synaptic weights are such that each node responds to its preferred pattern, and gives negligible responses to overlapping patterns. A novel input pattern consisting of two of the training patterns causes an incorrect response.

dynamical implementation would ensure that overlapping patterns could be distinguished after the saturation of synaptic weights, evenly divide activation between nodes which equally well match an input pattern, and perform exclusive allocation by providing competition for all inputs not just those represented by the winning node. In addition, a dynamical implementation would reduce quantisation error (and allow interpolation between preferred inputs) in a coarse coded representation (see section 6.2.1.1). However, the implementation adopted here does choose a winning node based on feedforward activation only since this is effective for most problems and is significantly less computationally expensive.

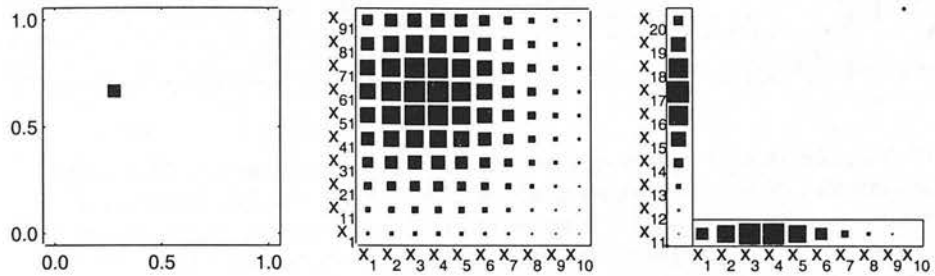
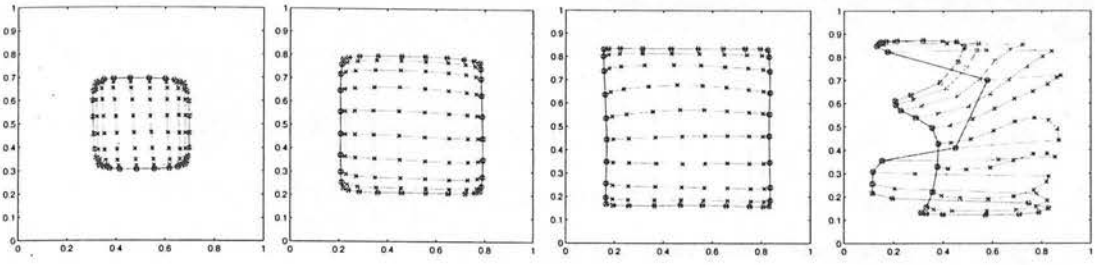


Figure 5.19: Generation of input data representing a point on the 2-dimensional plane. (a) The unit square of the 2-dimensional plane showing a data point. (b and c) Training data representing that data point (*i.e.*, the x_j values which form the input to the network). Each square represents an input source with the activation of this input proportional to the size of the square. The data is coarse coded so that each input is most strongly active for a preferred data point, but is broadly tuned to respond to data points near the preferred value with the strength of activation in proportion to the similarity of the data to the preferred value. (b) Using 100 input sources which are each selective for both the x- and y-coordinates of the data point. (c) Using two 10 input arrays one of which is selective to the x-coordinate and the other is selective to the y-coordinate of the data point.

5.3 Coarse and Topological Coding

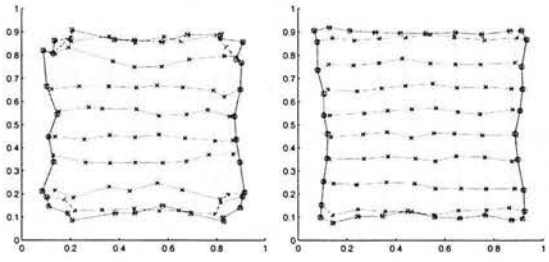
The results in the previous section have been generated using a network in which the winning node inhibits all other nodes equally (equation 4.14). By modulating the strength of lateral inhibition as a function of the distance between nodes (as described by equation 4.15) it is possible to form a topological representation of the input. To demonstrate that the algorithm can generate topologically ordered representations, an identical algorithm to that used to generate the above results has been used except for changes to the values of σ_E and σ_I ($\sigma_I = \frac{1}{3}l, \sigma_E = \frac{1}{3}\sigma_I$, where l is the maximum distance between any two nodes in the region).

A simple problem is to learn a topological map of data uniformly distributed on a 2-dimensional plane. In this application the input to the network is a coarse coded representation of the coordinates of a point on the plane. There are two possible methods of generating this input. Either the point is represented by a 2-dimensional array of values, or as two one-dimensional arrays representing the x- and y-coordinates of the point separately (figure 5.19). Figure 5.20(a) shows the result of using the first method of coding the input, while figure 5.20(b) shows the equivalent result generated using the second method. An additional way in which the input encoding affects the mapping is the choice of the width, σ_D, for the Gaussian which provides the coarse coding of the input data (this determines the size of the active population of inputs). Reducing this width can improve the accuracy of the mapping (figure 5.20(c)) since it makes different input values more distinct. This leads to nodes becoming more selective (more narrowly tuned) which in turn leads to the number of active nodes (the width of the output) becoming smaller. However, further reduction in the width of the input encoding can reduce the similarity between inputs to such an extent that the mapping is no longer topologically organised



(a) (b) (c) (d)

Figure 5.20: Topological maps of 2-dimensional data using weight decoding. A 10x10-node network was trained with data uniformly distributed over the unit square of the 2-dimensional plane. Maps of preferred inputs, measured by decoding the synaptic weights of the nodes, are shown after training with 4000 input patterns. (a) Data points were coded using a single 20x20 array of inputs to represent x- and y-coordinates (similar to figure 5.19(b)). (b) to (d) Data points were coded using separate 20x1 input arrays to represent x- and y-coordinates (similar to figure 5.19(c)). The width of the input encoding, σ_D, is the same in (a) and (b) $\sigma_D = 0.3$, while it decreases from (c) $\sigma_D = 0.15$, to (d) $\sigma_D = 0.1$.



(a) (b)

Figure 5.21: Topological maps of 2-dimensional data using activation decoding. A 10x10-node network was trained with data uniformly distributed over the unit square of the 2-dimensional plane. Maps of preferred inputs, measured by recording the response properties of the nodes, are shown after training with 4000 input patterns. Each input for which a particular single node is the most active is shown as a dot. Data points were coded using separate 20x1 input arrays to represent x- and y-coordinates (similar to figure 5.19(c)). The width of the input encoding is twice as large in (a) as in (b) and these are the same widths as those used in figure 5.20(b) and (c) respectively.

(figure 5.20(d)).

Results are plotted as maps showing each node projected onto the input plane at the location of its preferred input, with the preferred inputs of neighbouring nodes linked by lines (the preferred input of each node is shown using a cross and those of nodes at the edge of the map are also shown using circles). The preferred inputs may be found either by decoding the synapses or by decoding the activations (see section 4.1.3). Decoding the synaptic weights is done by taking the normalised sum of the synaptic weights weighted by the preferred coordinates of each input. The preferred input of node i is: $Pref_i = \frac{\sum_{j=1}^{m} q_{ij} Pref_j}{\sum_{j=1}^{m} q_{ij}}$, where q_{ij} is the weight of the synapse from input j, and $Pref_j$ is the preferred stimulus of input j (for a source receiving sensory data, as is the case here, $Pref_j$ is the physical stimulus that causes the highest value of x_j). All the maps in figure 5.20 were generated using this weight decoding method. Decoding the activations to measure the preferred input for a node is performed by observing for which inputs that node is the most active. The preferred input is the average of those stimuli,

spanning the entire input space, for which the node is the most active: $Pref_i = \texttt{mean}_t\{Stim_t\}\ \forall$ iterations t at which $y_i = \hat{y}$, where $Stim_t$ is the physical stimulus at iteration t. Maps generated using the preferred inputs measured in this way are shown in figure 5.21. These maps should be compared with those generated by decoding the synaptic weights for the same experimental conditions, which are shown in figures 5.20(b) and (c) (the increased accuracy due to reducing the width of the input encoding can be seen by comparing figures 5.21(b) and (a)). Both decoding methods bias the calculation towards the average stimulus, and this bias is increased by increasing the width of the input encoding. Consider a node representing a coordinate near the edge of the range, then this node cannot have weights to (non-existent) inputs representing values outside the range, but it will have weights to inputs representing values within the range, and hence its preferred stimulus, measured from weight decoding, will be moved away from the position of the peak weight towards the centre of the input distribution. It can be seen that measuring the preferred input by decoding the activation of the node is less biased towards the average input. However, the nodes still do not represent the edges of the input space as a node on the edge of the map will occasionally respond to inputs nearer the centre and these responses are unbalanced by responses to inputs outside the input space (which never occur). This method of decoding is more time consuming and all subsequent results are given for decoding the synaptic weights since these results are qualitatively similar. All subsequent figures have also been generated using an input encoding equivalent to that used in figure 5.20(b); a relatively wide input encoding and separate input arrays to represent x- and y-coordinates.

A network of 100 nodes arranged on a 10 by 10 square lattice was trained with data uniformly distributed over the unit square of the 2-dimensional plane. Figure 5.22 shows the map at different stages of development. It can be seen that the map becomes larger as RFs form and are differentiated by increased competition from the lateral inhibition. Once formed the map is very stable.

Figure 5.23(a) shows typical synaptic weight values for nodes in a network which forms a map of the 2-dimensional plane. The network was trained using input data coding for the x- and y-coordinates separately, the first 20 inputs receiving coarse coded data for the x-coordinate value and the second 20 synapses connected to inputs representing a coarse code of the y-coordinate value. It can be seen that nodes have become selective to a particular set of coordinates, but are broadly tuned. Nodes thus respond to a range of inputs, with the size of the response increasing towards the preferred input. Hence, the output of the network is also coarse coded. Figure 5.23(b) shows typical output activations for nodes in a network.

5.3.1 Robustness

As with local coding we require that the representations generated for coarse coding are robust to changes in the number of nodes in the networks. Figures 5.24(a) and (b) show the results, for net-

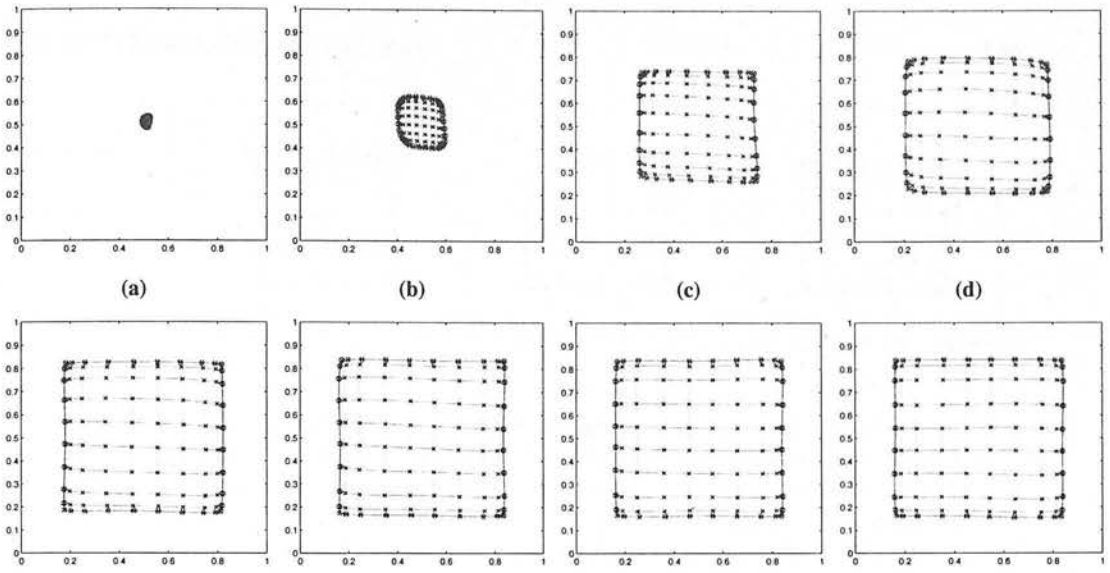


Figure 5.22: The development of a topological map. A 10x10-node network was trained with data uniformly distributed over the unit square of the 2-dimensional plane. Data points were coded using separate input arrays to represent the x- and y-coordinates. Maps of preferred inputs, measured by decoding the synaptic weights of the nodes, are shown after training with (a) 500, (b) 1000, (c) 2000, (d) 4000, (e) 8000, (f) 16000, (g) 32000, and (h) 64000 input patterns.

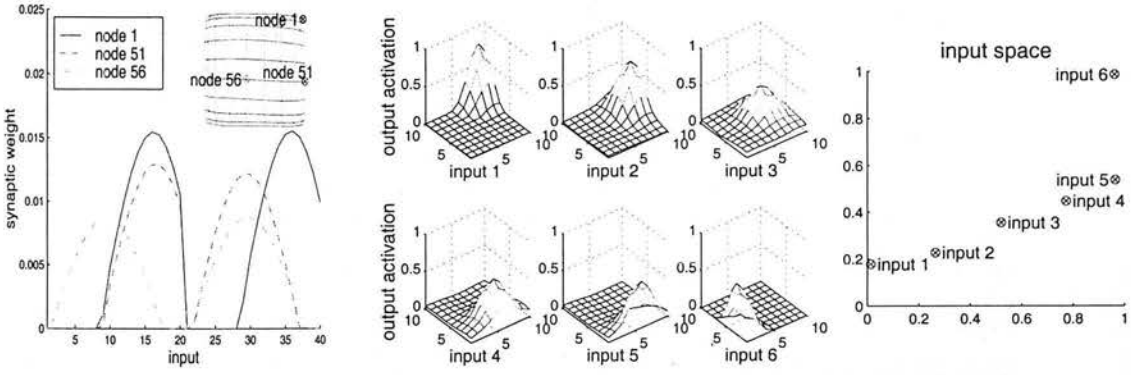
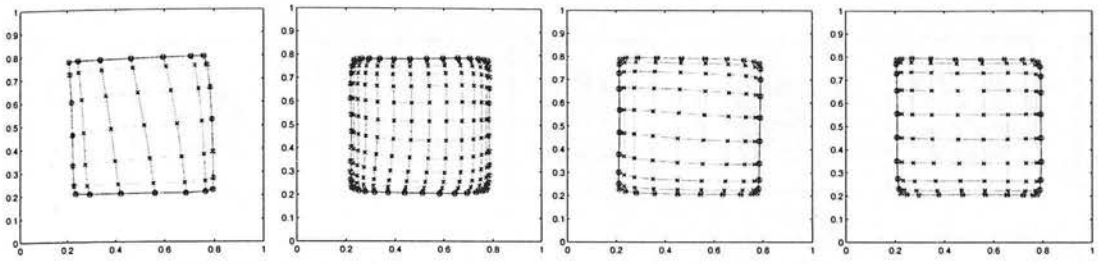


Figure 5.23: Coarse coding of input and output signals. (a) Typical receptive fields of nodes in a topological map of the unit square of the 2-dimensional plane after 4000 iterations. The synaptic weights are shown for 3 nodes whose positions are indicated on the inset. The 1st 20 inputs ($x_1 - x_{20}$) are supplied with a coarse coded representation of the x-axis coordinate value, while the 2nd 20 inputs ($x_{21} - x_{40}$) represent the y-axis coordinate value. (b) Typical output activations (y^{out}) for nodes in the same network. These activations are generated in response to the inputs shown on the right.



(a) (b) (c) (d)

Figure 5.24: Topological maps with varying numbers of nodes and numbers of inputs. Simple maps of the unit square of the 2-dimensional plane generated after training with 4000 input patterns. For networks containing varying numbers of nodes (a) 49 nodes, and (b) 196 nodes. And for networks receiving varying numbers of inputs (c) 80 inputs, and (d) 160 inputs. These results should be compared with the map generated for a network of 100 nodes receiving 40 inputs and trained for 4000 iterations shown in figure 5.22(d).

works with (approximately) half as many and twice as many nodes as used above, trained using the same data and the same parameter values. It can be seen that the algorithm is robust to changes in the number of nodes in the network. In addition, the network needs to be robust to different numbers of inputs (such input might be the output of a previous region which may have an arbitrary number of nodes). Figures 5.24(c) and (d) show the results for networks trained with the same data as before encoded using double and quadruple the number of input sources (fewer input sources than the original gave poorer results due to the poor resolution of the resulting training data). Again the algorithm is robust. This is confirmed by the result shown in figure 5.20(a) which was obtained using 2.5 times as many input sources as well as a different form of encoding for the input.

Scaling lateral inhibition to encourage topological map formation only provides a weak constraint on the network organisation. However, the algorithm proved to be reasonably robust. Previously, values were given for the robustness of networks (coding for horizontal and vertical bars) to random changes to the parameter values. For the problem being considered here such quantitative results are not given; instead the quality of maps produced using random parameter values is shown in figure 5.25. This figure shows 54 examples of maps generated using random parameter values chosen in the range $\pm 50\%$ from the hand-picked values used for all other results. It can be seen that most of these results provide a reasonable topologically organised map of the unit square. The algorithm would thus seem to be robust to changes in parameter values for this test case also.

The network proved to be robust to noise in the input data. Adding independent random noise, uniformly selected from the range $-v$ to $+v$, to each input source (and truncating the resulting input activations to be in the expected range of 0 to 1) caused the resulting map to be less square, but still allowed a reasonable representation of the 2-dimensional unit square to be formed, for all values of v (0.1, 0.25, 0.5, 0.75 and 1.0) that were tested.

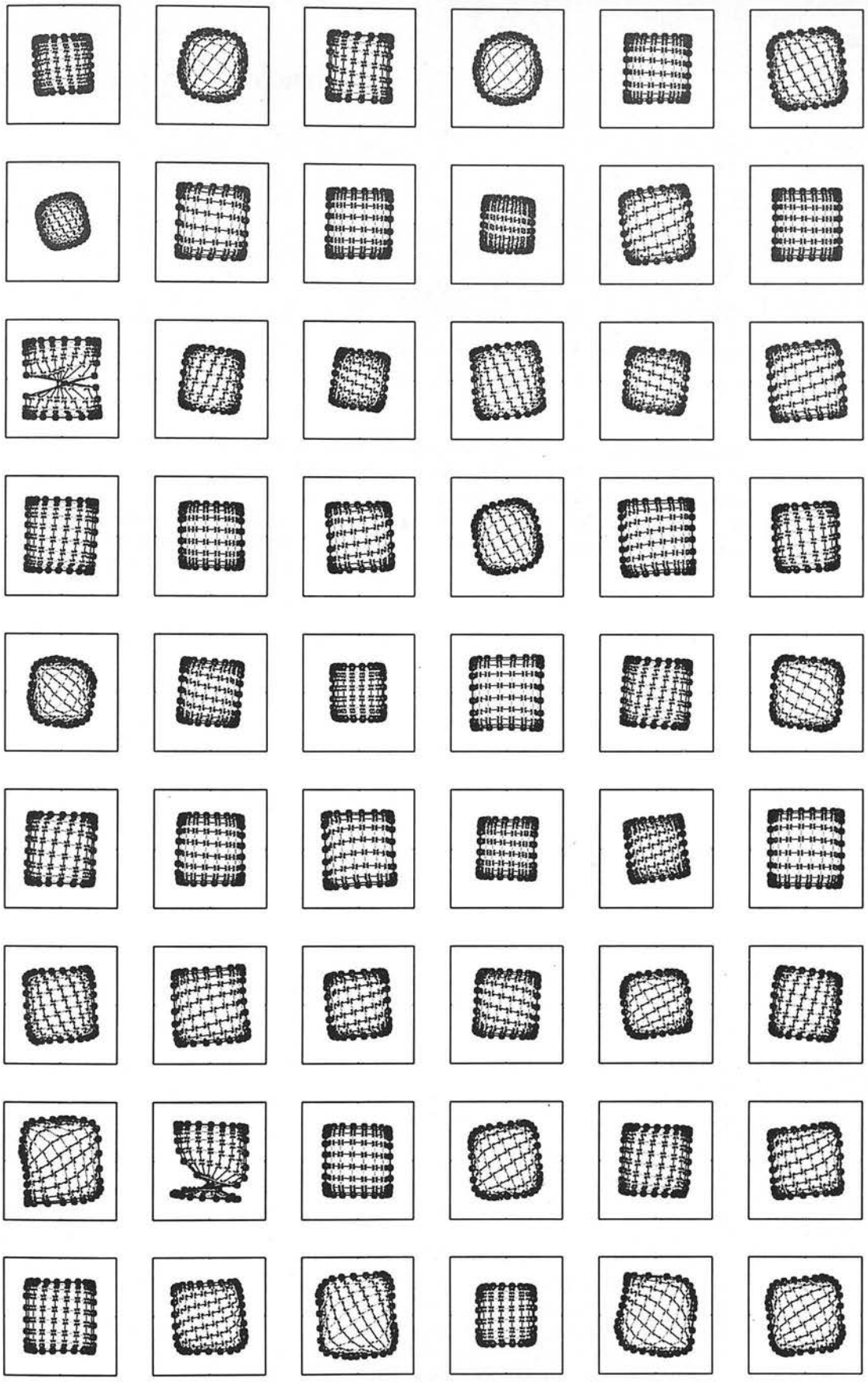
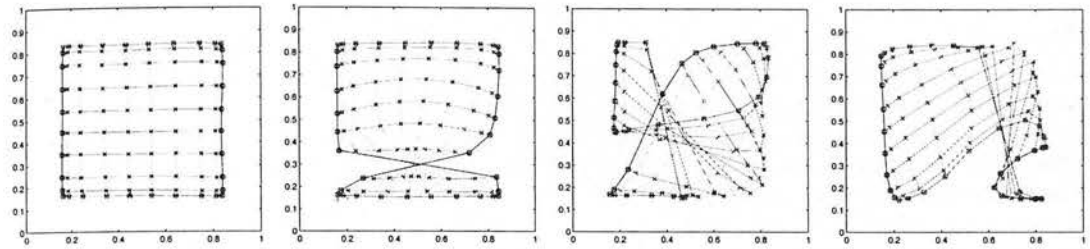
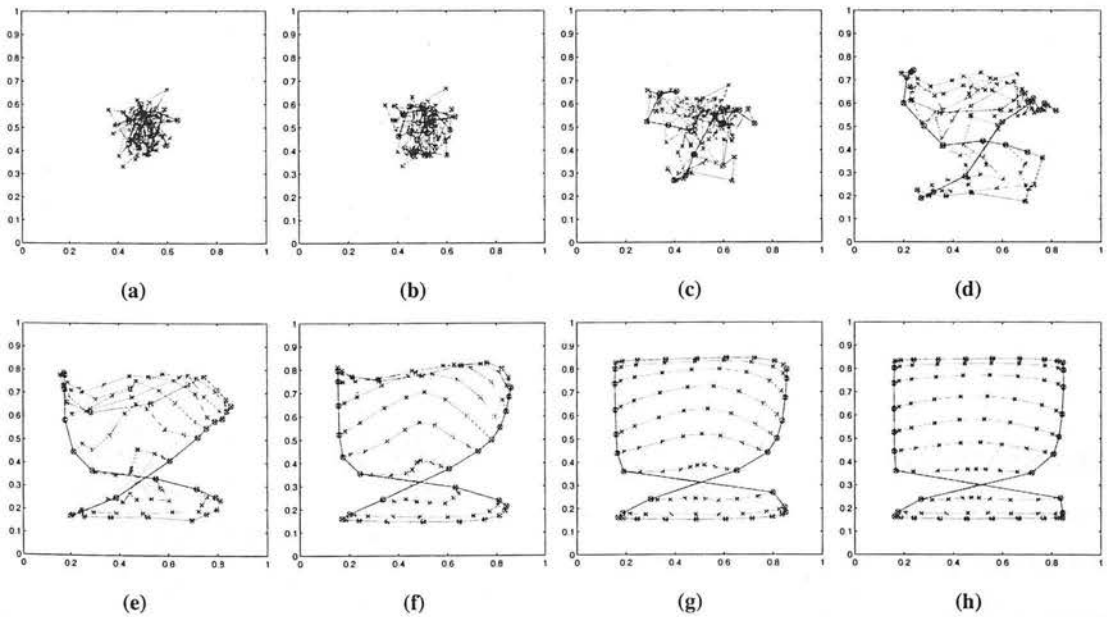


Figure 5.25: Topological maps with random changes in parameter values. Simple maps of the unit square of the 2-dimensional plane generated after training with 4000 input patterns. The network, input encoding, and decoding method is identical to that used to generate figure 5.22(d) but random parameter values are used. Parameter values have been uniformly chosen in the range $\pm 50\%$ from the hand-picked values used to generate figure 5.22(d).



(a) (b) (c) (d)

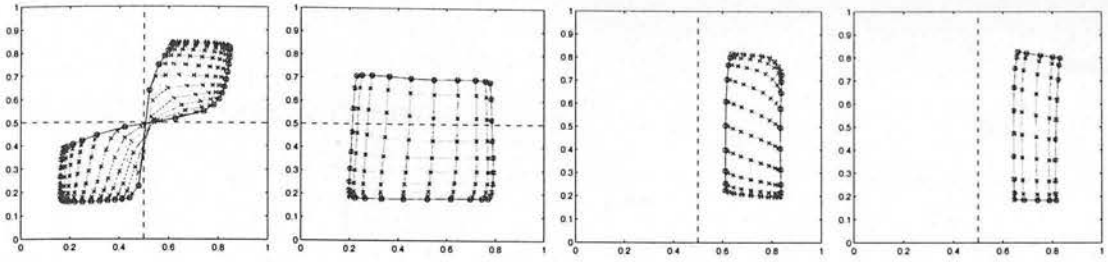
Figure 5.26: Topological maps with random initial synaptic weights. A 10x10-node network was trained with data uniformly distributed over the unit square of the 2-dimensional plane. Data points were coded using separate input arrays to represent the x- and y-coordinates. Maps of preferred inputs, measured by decoding the synaptic weights of the nodes, are shown after training with 64000 input patterns. The synaptic weights were initialised to have random values uniformly distributed in the range (a) 0 to 0.001, (b) 0 to 0.01, (c) 0 to 0.1, and (d) 0 to 1.0.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 5.27: The development of a topological map with random initial synaptic weights. A 10x10-node network was trained with data uniformly distributed over the unit square of the 2-dimensional plane. Data points were coded using separate input arrays to represent the x- and y-coordinates. Maps of preferred inputs, measured by decoding the synaptic weights of the nodes, are shown after training with (a) 500, (b) 1000, (c) 2000, (d) 4000, (e) 8000, (f) 16000, (g) 32000, and (h) 64000 input patterns. The synaptic weights were initialised to have random values uniformly distributed in the range 0 to 0.01.



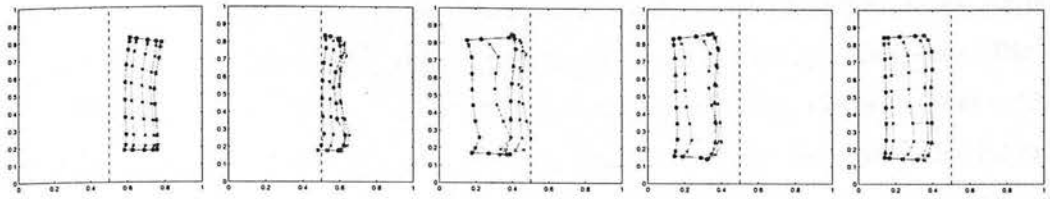
(a) (b) (c) (d)

Figure 5.28: Topological maps of other 2-dimensional data. Maps of preferred inputs, measured by decoding the synaptic weights of the nodes, are shown after training with 4000 input patterns. Data points were coded using separate input arrays to represent the x- and y-coordinates. (a) A 20x10-node network was trained with data uniformly distributed over the lower-left and upper-right quadrants of the unit square of the 2-dimensional plane. (b) A 10x10-node network was trained with data distributed over the unit square of the 2-dimensional plane such that the density of data points in the lower half of the plane was three times that in the upper half of the plane. (c) A 10x10-node network was trained with data uniformly distributed over the right side of the 2-dimensional plane. (d) A 10x5-node network was trained with data uniformly distributed over the right side of the 2-dimensional plane. Note that in all cases nodes receive synapses from input sources representing all coordinates, as before.

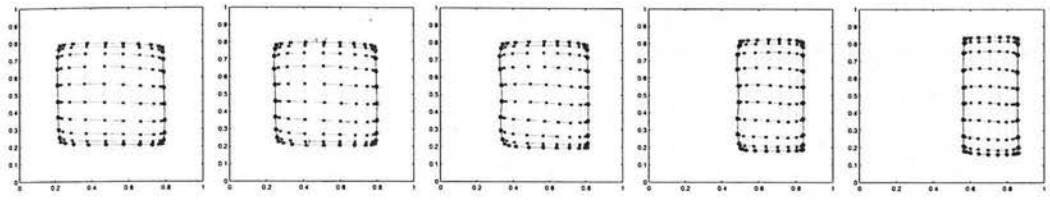
The algorithm proved to be reasonably robust to initial random synaptic weights. Initialising the synaptic weights to be random values uniformly selected from the range 0 to v caused a poor topological organisation for values of v greater than 0.001 (see figure 5.26). For $v = 0.001$ the average synaptic weight is 0.0005. Since the average increase in each synaptic weight at each iteration is given by $\beta = 0.00000225$ this average initial synaptic weight is comparable to the weight generated by 220 training cycles. For v greater than 0.001 the randomness in the initial weights led to discontinuities in the map formed. However, the receptive fields of the network still provided coverage of the input space even though they were not well organised topologically. The formation of a representation that covered the input space was slowed down by increasing the size of the random initial weights. Figure 5.27 shows the development of a map for which the initial synaptic weights are random values uniformly selected from the range 0 to 0.01. Since each node has 40 synapses this corresponds to each node having a total synaptic weight of 0.2 at the start.

5.3.2 Stability

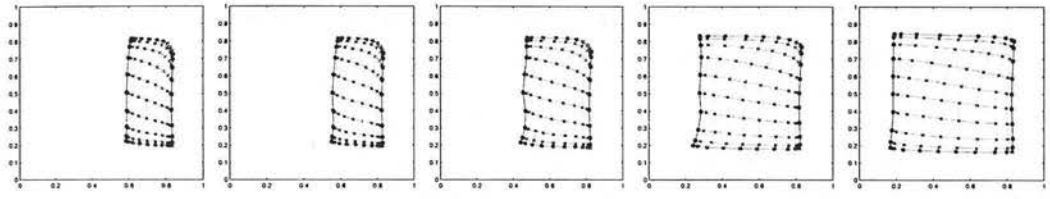
Topological maps of other 2-dimensional distributions can be learnt. Figure 5.28(a) shows a map in which input data is uniformly distributed in only two quadrants of the unit square. Because receptive fields are developed through activity dependent modification (rather than through the use of a 'neighbourhood' function, see section 4.3.2.4) the network successfully represents discontinuous input distributions. In figure 5.28(b) the density of data points in the upper and lower halves of the unit square are different with training data occurring in the lower half with three times the frequency that data occurs in the upper half. Since habituation keeps nodes winning with approximately equal frequency there is a



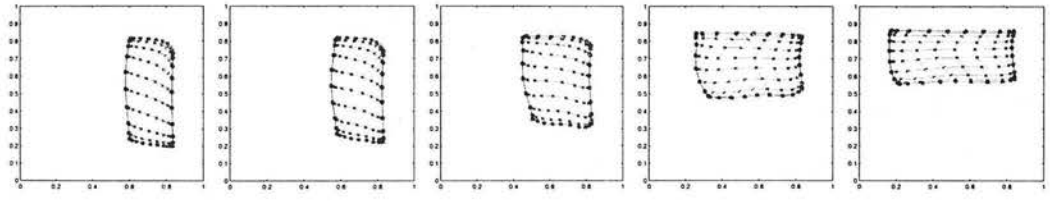
(a)



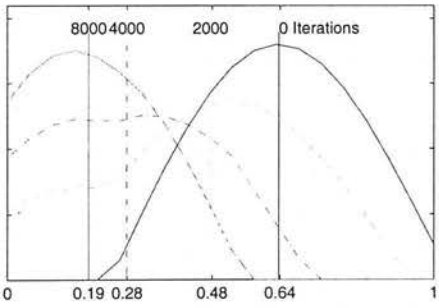
(b)



(c)



(d)



(e)

Figure 5.29: Topological maps after changes in the input distribution. (a) A 10x5 network trained for 4000 iterations on data from the right half plane (as in figure 5.28(d)) is subsequently trained on data from the left half plane only. Results are shown after training on the new input distribution for 1000, 2000, 4000, 8000 and 16000 iterations. (b) A 10x10 network trained for 4000 iterations on data from the full unit square (as in figure 5.22(d)) is subsequently trained on data from the right half plane only. Results are shown after training on the new input distribution for 500, 1000, 2000, 4000 and 8000 iterations. (c) A 10x10 network trained for 4000 iterations on data from the right half plane (as in figure 5.28(c)) is subsequently trained on data from the full unit square only. Results are shown after training on the new input distribution for 500, 1000, 2000, 4000 and 8000 iterations. (d) A 10x10 network trained for 4000 iterations on data from the right half plane (as in figure 5.28(c)) is subsequently trained on data from the top half plane only. Results are shown after training on the new input distribution for 500, 1000, 2000, 4000 and 8000 iterations. (e) Shows the change in the synaptic weights, connected to inputs representing one coordinate value, of a typical node. The vertical lines indicate the decoded value of each weight array, after training with the second input distribution for 0, 2000, 4000, 6000 (unlabelled), and 8000 iterations.

higher density of nodes representing the densest part of the input distribution and hence the resolution of the mapping is appropriate to the input distribution. In figures 5.28(c) and (d) input data uniformly distributed over half the unit square was used. This input data was encoded in the same way as before.

The half-plane input data was used, together with the full-plane data, to test the plasticity of the mappings. Networks were trained for 4000 iterations using one input distribution. The input was then suddenly changed to another distribution and training continued. Figures 5.29(a) to (d) show the development of maps after switching the input distribution. These results show that the network remains plastic to changes in the input distribution. The length of training required to reorganise the receptive fields is longer than the time that would be required to generate the maps directly. This is due to the slow reduction in the strength of the original weights: until these residual weights are removed the decoding of the synaptic weights is biased towards the original preferred input (figure 5.29(e)). In situations where a more profound change in the topology of the input space occurs (*e.g.*, a change in its shape which is not a simple transformation of the original shape, or a change in the dimensionality of the space) the original weights may prevent a good map forming of the new data. In such cases the original input weights act rather like randomly initialised weights and cause the resulting map to be badly organised or discontinuous.

All the results in this section have considered a continuous input space. It would be possible to encode the position of a point on the 2-dimensional plane with a discontinuous representation by having only one input source active at a time (*e.g.*, an encoding like that shown in figure 5.19(a)). In this case there would be no overlap between the representations of adjacent points in the plane and a network receiving this input data could not form a topological representation (although given a network with at least as many nodes as there were input sources it would be able to represent the entire input space). It is, however, possible to form topological maps of discrete input spaces, provided there is still overlap between the distinct input patterns. Consider a network of six nodes receiving data from six input sources labelled 'a', 'b', 'c', 'd', 'e', and 'f' (see figure 5.17). If the network is trained with six input patterns 'a', 'ab', 'abc', 'cd', 'de', and 'def' then the network will form a topologically ordered representation in which neighbouring nodes represent inputs patterns which overlap (figure 5.30, this is in contrast to the unordered representation formed without modulation of the lateral inhibition strength, figure 5.18). A topological representation of a continuous input space is coarse coded. That for a discrete input space is not; each node has strong weights to its preferred input pattern only.

5.4 Learning Synaptic Clustering

All the results in previous sections have used linear nodes. This section experiments with using nodes with a nonlinear combination function, as described in section 4.4.2. Since nodes can form multiple

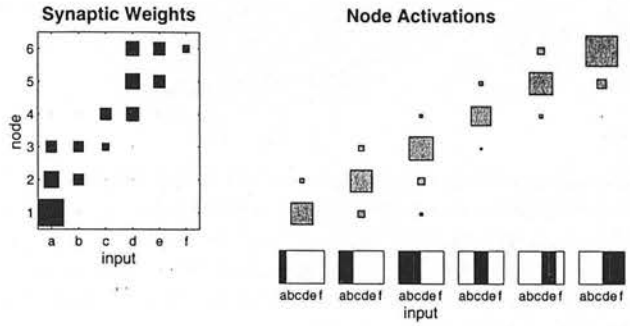
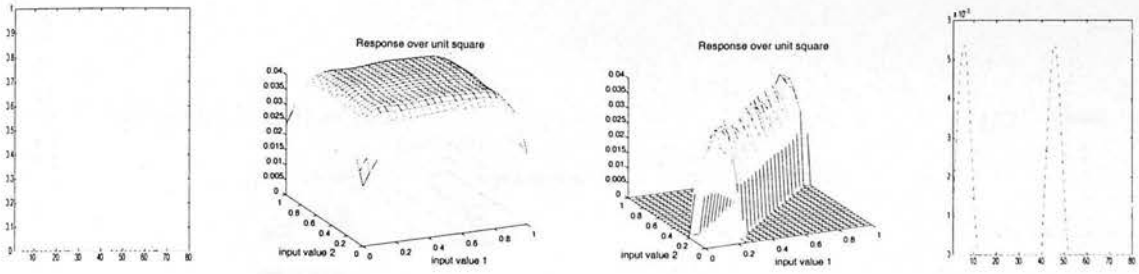


Figure 5.30: Learning overlapping input patterns with topological ordering. Six nodes are trained using input patterns 'a', 'ab', 'abc', 'cd', 'de', and 'def'. Results are shown after 4000 training cycles. The strength of synaptic weights between each node and each input are shown on the left using squares which have sides of length proportional to the synaptic weight. Input patterns are illustrated along the bottom of the right side, active inputs being represented by dark squares. The average response to these input patterns of each node over a number of trials is shown on the right using squares which have sides of length proportional to the node activations. Each node learns to represent one of the input patterns. The synaptic weights are such that each node responds to its preferred pattern, and gives a small response to overlapping patterns, and no response to distinct patterns.

clusters of inputs it is necessary that the learning rules for the synaptic weights are modified in order to prevent any reduction in synaptic weight when an input is inactive. This means that nonlinear nodes do not refine their receptive fields as well as linear nodes. This is particularly a problem when these weights are used to provide lateral inhibition since nodes will partially inhibit inputs which are not their preferred inputs. Hence, due to the difference in synaptic learning rule, nonlinear nodes fail to solve the bars problem.

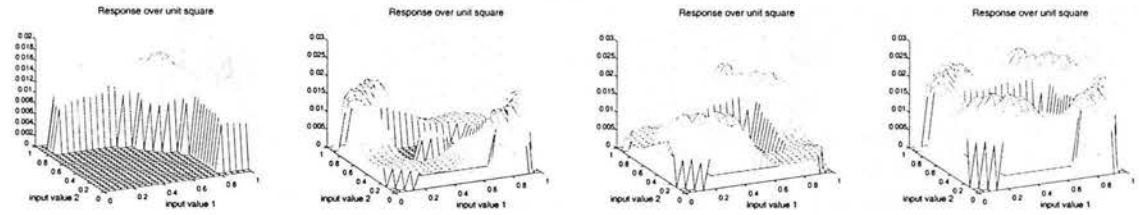
5.4.1 Continuous Input

Consider the problem of learning to represent (using one node) points along the leading diagonal of the unit square of the 2-dimensional plane. The two coordinates of the input space were represented by separate, coarse coded, arrays (as in figure 5.19(c)). Within each of these arrays the coordinate of a data point is represented by a Gaussian activity distribution centred on the coordinate value (figure 5.31(a)). Training data consisted of uniformly distributed randomly selected points along the diagonal. Since this training data spans the entire range of each coordinate all synaptic weights become equally strong, and hence, without using any synaptic interactions (by keeping $c_{jk} = 0 \; \forall j, k$) the node responds, after training, equally to any point within the unit square (figure 5.31(b)). However, when the learning of cluster weights is allowed they are formed between synapses representing corresponding coordinate values (figure 5.31(d)). Hence, a node using synaptic interactions responds to test data along the leading diagonal only (figure 5.31(c)). Similar results can be generated for learning to represent arbitrary curves. Figure 5.32(a) shows the response of a node trained to represent the circumference of a quadrant of a circle. To represent the entire circumference of a circle would require a synapse representing a particular



(a) (b) (c) (d)

Figure 5.31: Learning to represent points along the leading diagonal of a 2-dimensional input space. (a) Training data consisted of two population coded input arrays, representing x- and y-coordinates on the 2d plane. This data was supplied to the node such that the first 40 synapses received the population coded representation of the x-coordinate, and the second 40 synapses received the y-coordinate value. Input data representing two points along the diagonal are shown (each in a different line style). (b) and (c) The response of the node after training. The strength of the response of the node which is caused by input data representing points across the unit square, is shown for (b) a linear node, and (c) a nonlinear node ($\kappa = 0.0005$). (d) The cluster weights that have been learnt for four synapses (cluster weights for each synapse are in a different line style). The horizontal line shows the threshold (κ).



(a) (b) (c) (d)

Figure 5.32: Learning to represent points along a curved line of a 2-dimensional input space. Each figure shows the response of a nonlinear node after training. The strength of the response of the node which is caused by input data representing points across the unit square is shown. (a) For a single node trained with data along the circumference of a quadrant of a circle of radius 1 and centre at the origin. (b) and (c) The response of two, competing nodes, trained with data along the circumference of a circle. (d) The sum of the responses in (b) and (c) representing the response of the virtual nonlinear unit composed of these two nodes. ($\kappa = 0.0005$.)

x-coordinate (or y-coordinate) to form clusters with inputs representing two distinct y-coordinate (x-coordinate) values. Since each synapse has only a single cluster weight vector it can only take part in a single cluster (section 4.4.2), but the output of multiple, competing, nodes can act as a larger virtual unit. Figures 5.32(b) and 5.32(c) show the response of two nodes which compete to represent points on the circumference of a circle. It can be seen that each node has come to represent diagonally opposite quadrants of the circle, where each synapse need only take part in a single cluster. The sum of these two responses represents the response of a virtual unit composed of these two nodes and is shown in figure 5.32(d).

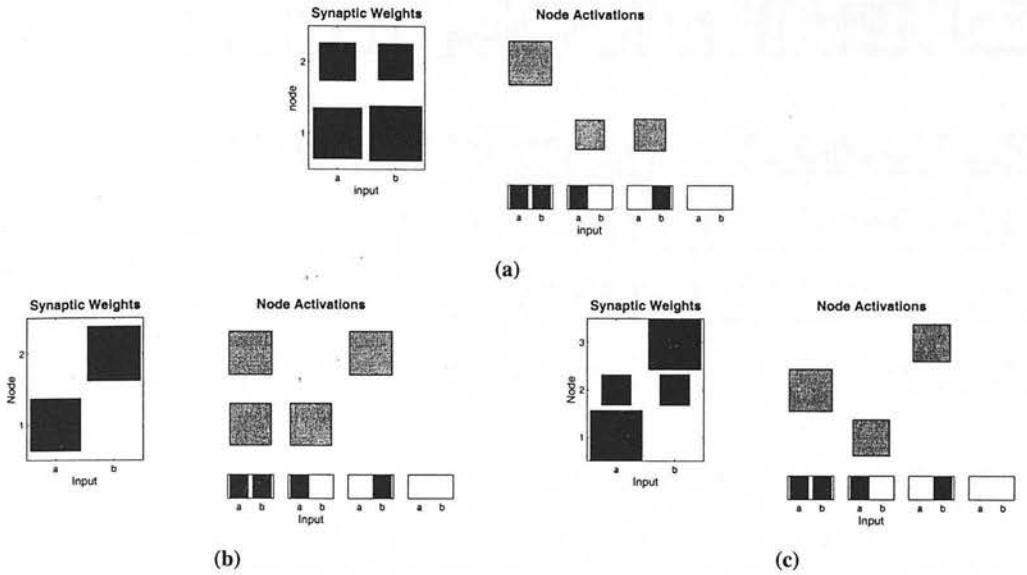


Figure 5.33: Learning the XOR function. (a) A network of two nonlinear nodes represents the AND and the XOR of the two inputs ($\kappa = 0.00001$). (b) A network of two linear nodes represents each input separately. (c) A network of 3 linear nodes represents each input separately and the AND of the two inputs.

5.4.2 Discontinuous Input

A standard problem to which nonlinear nodes are applied is to represent the XOR function. It is well known that XOR can be detected using an AND function in combination with an OR function. A network of two nonlinear nodes can produce such a detector (figure 5.33(a)). Both nodes form strong synaptic weights to both inputs. However, one node forms strong cluster weights between the synapses and comes to respond only to the combination of both inputs. The other node does not cluster inputs and will respond to either input, however, it is prevented from responding when both inputs are active by the lateral inhibition of the other node. This node thus responds only to the XOR function. In contrast a 2-node network of linear nodes is unable to detect the XOR function. Each linear node represents an individual input (figure 5.33(b)). With additional linear nodes ones which respond to the AND of both inputs will occur in addition to those representing the individual inputs (figure 5.33(c)). However, no node representing the XOR of the inputs can form as this requires strong connections to both inputs and will thus respond to AND. A second region of linear nodes receiving input from a network representing 'a', 'b', and 'a AND b' could solve the XOR problem.

Cluster learning has been successfully applied to a discontinuous, binary valued, input space (learning the XOR function) and to a continuous, real valued, input distribution (learning to represent lines on a plane).

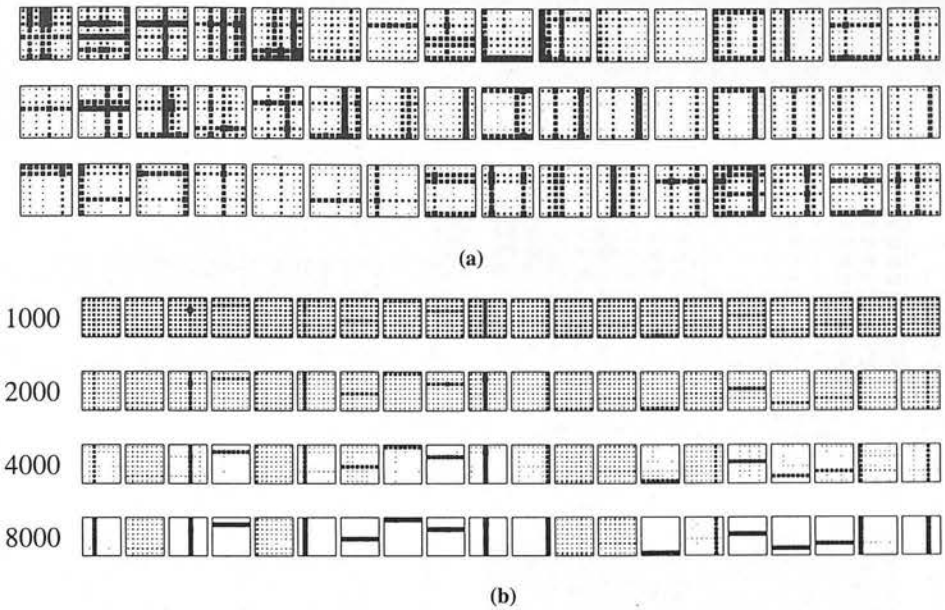


Figure 5.34: Using sensitisation when there is no temporal correlation in the input data. (a) 48 typical input patterns for the bars problem when using a sensitisation factor of $\lambda_x = 0.5$. There is no noise in the input data, but previously active inputs remain partially active for subsequent iterations. (b) The synaptic weights for the 20 nodes in a network after training on 1000, 2000, 4000 and 8000 input patterns.

5.5 Learning Invariant Representations

In all the examples above training data has consisted of randomly generated inputs, in which there is no correlation between one input pattern and the next. The sensitisation factor, λ_x, has thus been set to a value of one (when $\lambda_x = 1.0$ there is no sensitisation of the current input to previous values). However, it is possible to use sensitisation even when there are no temporal correlations between patterns. In this case the persistence of the previous input values caused by sensitisation acts rather like noise, but since the algorithm is robust to noise it succeeds in generating suitable representations under these conditions (*e.g.*, figure 5.34). This section considers using sensitisation ($\lambda_x = 0.5$), when there is temporal correlation in the input data, in order to learn translation invariant representations. Input data is provided such that different views of the same input pattern occur in sequences of random length. There are no null inputs to separate sequences corresponding to different patterns and hence the input data does not provide any information about how to segment the input data into different classes.

5.5.1 Horizontal and Vertical Bars

Consider the problem of trying to distinguish (using a single layer of two nodes) horizontal and vertical bars on an 8 by 8 grid. Learning is unsupervised with the two nodes competing to represent inputs. Since there is no overlap between bars of the same orientation an extra constraint is required to cause

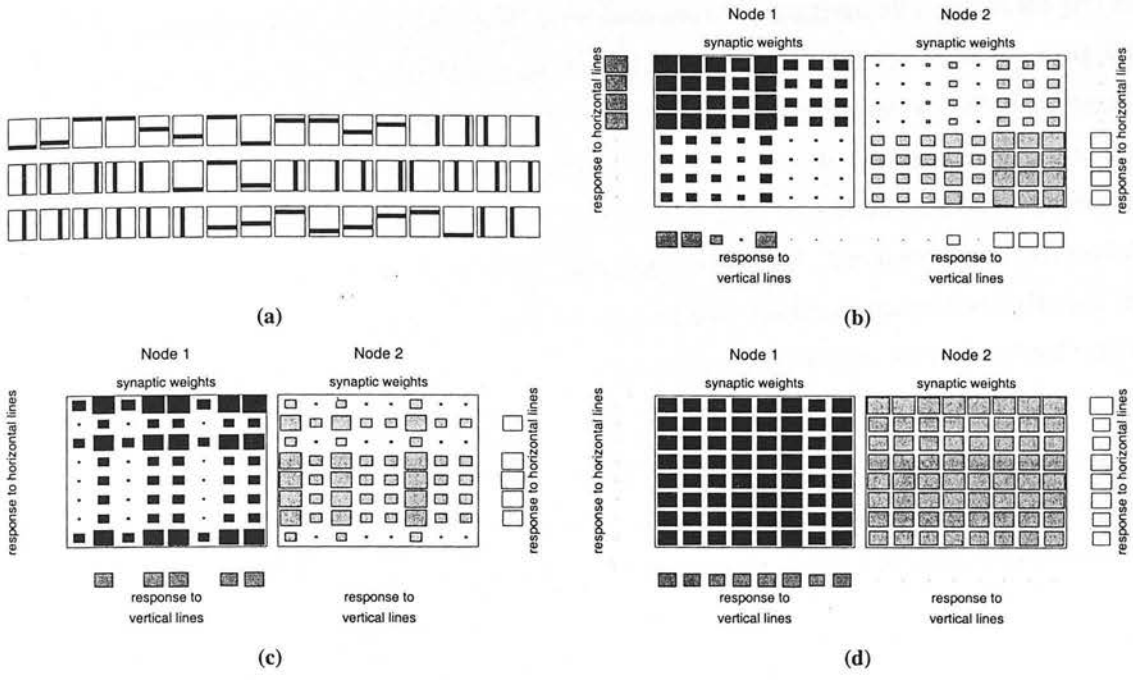


Figure 5.35: Learning to represent horizontal and vertical bars. Two nodes receive input from an 8 by 8 array of binary inputs. (a) Training data consists of one only of the 16 possible horizontal or vertical bars being active at any one time with short sequences of bars at the same orientation (a new, random, orientation is chosen with probability 0.25 at each iteration). (b), (c), and (d) The squares within each box labelled 'synaptic weights' illustrate the strengths of the synaptic weights connecting grid points to a node. These squares are proportional in size to the weight they represent. The squares along the edges of these boxes indicate the strength of the (mean) activation of that node on presentation of a bar at the corresponding position in the grid. These squares are proportional in size to the activation they represent. (b) Linear nodes with sensitisation. (c) Nonlinear nodes without sensitisation ($\kappa = 0.00025$). (d) Nonlinear nodes with sensitisation ($\kappa = 0.00025$). Cluster weights are not shown. However, in the final case, when nonlinear nodes are used with sensitisation, each synapse, receiving input from grid point g, forms strong cluster weights only to those synapses receiving input from one orientation of bar through grid point g, and all the synapses on the same node form clusters representing the same orientation of bars.

them to be classified together. One such constraint is to increase the probability that contiguous inputs are of the same orientation (figure 5.35(a)), and to use sensitisation. A similar approach has previously been applied to this problem (Templeman and Loew, 1989; Földiák, 1991), however those implementations used linear units resulting in it only being possible to learn invariances when there was little overlap between each set of translated patterns[7] (Becker, 1993) and the nodes will incorrectly respond if the input consists of partial patterns from different translational positions[8]. Using (a single layer) of two linear units could not solve the horizontal/vertical bars problem. A single layer of linear nodes

[7] To overcome this problem both Templeman and Loew (1989) and Földiák (1991) used a second layer of linear units to provide the input to the layer learning the invariance. This input layer consisted of nodes that responded to one orientation of bar only at each grid location. Hence, there was no overlap between any of the patterns supplied to the layer learning invariance.

[8] To overcome this problem Riesenhuber and Poggio (1999b) have proposed that cells which learn invariance should respond to the maximum input only rather than the sum of inputs.

divides the weight array arbitrarily so that each represents some horizontal and some vertical bars (figure 5.35(b)). A similarly arbitrary division of the input space also occurs when nonlinear nodes are used but with $\lambda_x = 1.0$ (figure 5.35(c)). However, since the synapses cannot participate in more than one cluster some bars are not represented at all. Since both sets of bars activate all points in the grid a node representing horizontal bars would have equal weights to all inputs, and a node representing vertical bars would have a similar weight array. Thus, the weight space alone can not be used to distinguish the bar's orientation. However, by using synaptic interactions together with sensitisation, nodes can learn to exclusively represent either vertical or horizontal bars (figure 5.35(d)). Both nodes learn similar weight arrays, but the cluster weights distinguish between bar orientations. The interference between cluster weights representing horizontal and vertical bars in each individual node means that nodes will tend to only learn to represent one orientation, and hence correct representations are also formed with lower values of sensitisation.

As well as demonstrating the use of sensitisation to learn from sequences of inputs this experiment has also used cluster learning. It is also possible to learn invariant representations without using non-linear nodes: section 6.1.2.1 solves an identical problem to that posed here using two layers of linear nodes, and the following example also uses linear nodes.

5.5.2 Corresponding Bars

Consider a network which receives two streams of input data (as shown in figure 5.1(b)). If the source of input to both streams is the bars data (figure 5.2) and this data is uncorrelated across the streams, then this constitutes a larger version of the bars problem, in which there are 32 independent bars on an 8 by 16 grid. A sufficiently large network (*i.e.*, one with at least 32 nodes) can learn to use individual nodes to represent each individual bar. However, here we consider the case where there is correlation between the active bars in each stream and nodes learn to represent corresponding bars from both input streams. If both data streams received identical input patterns simultaneously then finding correspondences between streams would be trivial: it would be a case of learning 16 larger patterns presented to the 8 by 16 grid. In this experiment patterns are not presented simultaneously but sequentially to each data stream. Correspondences are thus learnt as a result of temporal correlations only. A node which successfully learns to represent a pair of corresponding bars from each input stream can thus be considered to have formed a translation invariant representation of a particular bar across two locations.

The bars data was generated as before. Input patterns were supplied to alternating data streams in a sequence of random length (between 0 and 10 presentations) to a network of 16 linear nodes. For each presentation of the input pattern to one data stream the other stream received either a blank input or another random bars pattern. In this manner, identical patterns were not presented simultaneously to both data streams, but were presented to each data stream sequentially. Using multiple numbers of

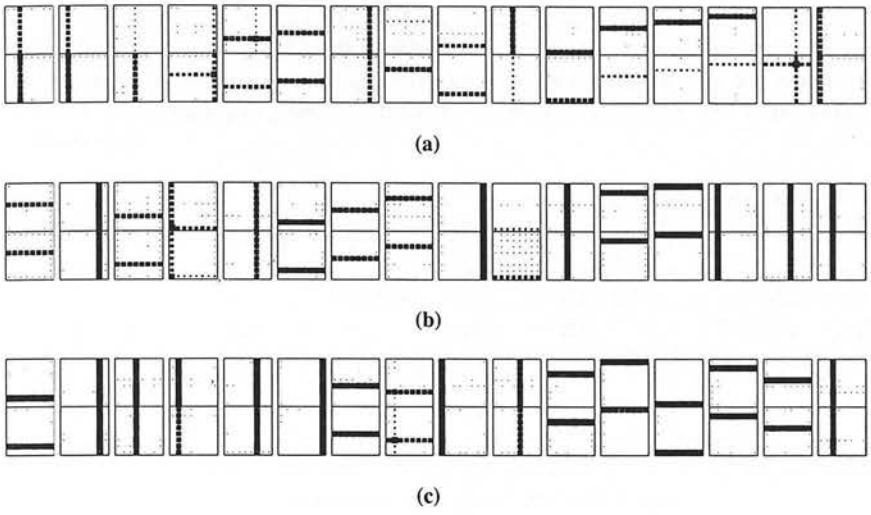


Figure 5.36: Learning to represent a specific bar invariant to translation. Input patterns containing bars on an 8 by 8 grid were supplied to each of two input streams converging on a single network of 16 nodes. The same pattern was presented to one data stream and then the other repeatedly for sequences of variable length. The strength of synaptic weights is shown such that weights to inputs in one data stream are above the weights to the other data stream. Results are shown after 4000 training cycles. (a) Multiple bars were active in the data, and the data stream not receiving the input patterns was supplied with another random bars pattern. Only a partially translation invariant representation is learnt. (b) Multiple bars were active in the data, and the data stream not receiving the input patterns was supplied with a blank image. (c) Single bars were active in the data, and the data stream not receiving the input patterns was supplied with another pattern containing a single active bar.

bars in each pattern and supplying a random bars pattern, also containing multiple bars, to the other data stream at each iteration it is very hard to learn a translation invariant representation (figure 5.36(a)). This is not surprising since many bars are active at each iteration and the sensitisation factor means that previous inputs are also partially active resulting in all inputs being active at each iteration and hence the corresponding inputs in each stream are very indistinct. The ability to learn a translation invariant representation was improved by supplying blank inputs to the other data stream at each iteration (figure 5.36(b)) or by using single active bars in each pattern (figure 5.36(c)). This problem is complex since it requires learning to perform abstraction in order to group together inputs that represent the same bar in one stream, and to perform invariance in order to represent multiple bars occurring at different times in different streams.

The use of sensitisation to learn invariant representations has been successfully applied to linear nodes (learning corresponding bars), and to nonlinear nodes (learning to distinguish horizontal and vertical bars).

input pattern	parity	response of one nonlinear unit	response of one linear unit
0 0	0	*0.0*	*0.0134*
0 1	1	*0.0133*	*0.0133*
1 0	1	*0.0135*	*0.0135*
1 1	0	*0.0*	*0.0134*

Table 5.2: Learning the XOR problem with one node. The first two columns give the input patterns. The last two columns give the mean output responses of nodes after training for 8000 iterations. The fourth column is the response of a nonlinear node ($\kappa = 0.00025$). The fifth column is the response of a linear node.

input pattern	parity	response of two nonlinear units			response of two linear units		
0 0	0	0.0	0.0	*0.0*	0.0099	0.0083	*0.0182*
0 1	1	0.0	0.0224	*0.0224*	0.0	0.0245	*0.0245*
1 0	1	0.0230	0.0	*0.0230*	0.0242	0.0	*0.0242*
1 1	0	0.0	0.0	*0.0*	0.0089	0.0094	*0.0183*

Table 5.3: Learning the XOR problem with two nodes. The first two columns give the input patterns. The last six columns give the mean output responses of nodes after training for 8000 iterations. The fourth and fifth columns are the responses of two nonlinear nodes, and the sixth column is the summed response of these two nodes ($\kappa = 0.00025$). The seventh and eighth columns are the responses of two linear nodes, and the ninth column is the summed response of these two nodes.

5.6 Learning with Apical Supervision

In section 4.2.4 a network was described as having two distinct sets of inputs: one received by the basal dendrites and a second received by the apical dendrites. In all the above applications there have been only connections to the basal dendrites. In this section the effect of connections to the apical dendrites is considered. Apical inputs do not affect the output of the nodes, but do affect the learning. Such inputs can thus be used to provide supervision, reinforcement, or bias for learning appropriate representations at the basal synapses, and for finding appropriate synaptic clusters within the basal dendrite. As for basal inputs, the apical input could come directly from the environmental or from the output of another region.

5.6.1 Supervised Learning

Consider using a single apical input to supervise learning to detect the parity of a binary input vector (equivalent to the XOR function when the input vector is 2-bits). This single apical input acts as a supervisory signal for the learning in the basal dendrites; it is supplied with the required parity value. In cases where multiple nodes act as a virtual unit, this supervisory signal specifies the required output of the virtual unit, rather than the outputs of individual nodes, and the nodes within the virtual unit compete (as in unsupervised learning) to divide the problem between them. Since the feedforward synaptic weights in this model are constrained not to be negative, it is necessary to use an encoding of the input vector containing each bit and its complement (such an encoding would be required to

input pattern	parity	response of one nonlinear unit	response of one linear unit
0 0 0	0	*0.0581*	*0.0581*
0 0 1	1	*0.0588*	*0.0588*
0 1 0	1	*0.0582*	*0.0582*
0 1 1	0	*0.0583*	*0.0583*
1 0 0	1	*0.0584*	*0.0584*
1 0 1	0	*0.0591*	*0.0591*
1 1 0	0	*0.0584*	*0.0584*
1 1 1	1	*0.0587*	*0.0587*

Table 5.4: Learning the 3-bit parity problem with one node. The first three columns give the input patterns. The last two columns give the mean output responses of nodes after training for 16000 iterations. The fifth column is the response of a nonlinear node ($\kappa = 0.00025$). The sixth column is the response of a linear node.

input pattern	parity	response of two nonlinear units			response of two linear units		
0 0 0	0	0.0019	0.0407	*0.0425*	0.0	0.0710	*0.0710*
0 0 1	1	0.0	0.0732	*0.0732*	0.0	0.0718	*0.0718*
0 1 0	1	0.0	0.0719	*0.0719*	0.0	0.0708	*0.0708*
0 1 1	0	0.0059	0.0374	*0.0433*	0.0040	0.0646	*0.0686*
1 0 0	1	0.0716	0.0	*0.0716*	0.0705	0.0	*0.0705*
1 0 1	0	0.0407	0.0022	*0.0429*	0.0709	0.0	*0.0709*
1 1 0	0	0.0407	0.0019	*0.0425*	0.0708	0.0	*0.0708*
1 1 1	1	0.0721	0.0	*0.0721*	0.0711	0.0	*0.0711*

Table 5.5: Learning the 3-bit parity problem with two nodes. The first three columns give the input patterns. The last six columns give the mean output responses of nodes after training for 16000 iterations. The fifth and sixth columns are the responses of two nonlinear nodes, and the seventh column is the summed response of these two nodes ($\kappa = 0.00025$). The eighth and ninth columns are the responses of two linear nodes, and the tenth column is the summed response of these two nodes.

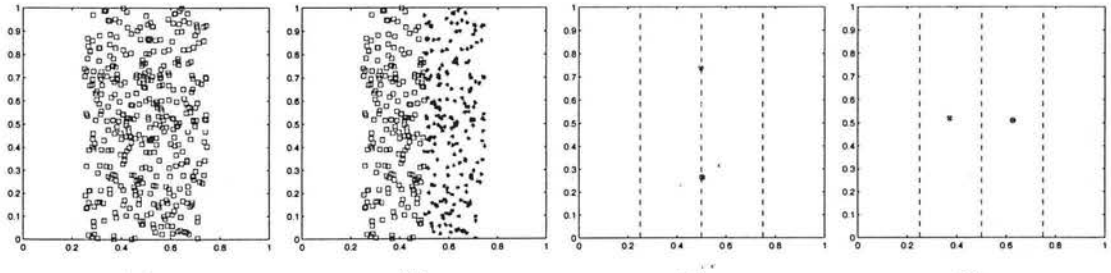
solve this problem by any other neural network with the same restriction on synaptic weights). The two bit parity problem (or XOR problem) can be solved using a single nonlinear node. A linear unit, trained in the same way, is unable to fully solve this problem. The mean responses of the nodes, after training on this problem, are shown in table 5.2. Two linear units are still unable to fully solve the XOR problem, while two nonlinear units (acting as a virtual unit) produce a good solution (table 5.3). For parity problems involving more bits each input needs to form a part of more than one cluster, so that solutions can only be found using multiple nodes acting as a larger virtual unit. Hence, using one unit, both linear and nonlinear nodes fail to solve the 3 bit parity problem (table 5.4, the results for linear and nonlinear nodes are identical as the conflicting requirements for the cluster weights in the nonlinear case cause all cluster weights to remain below κ). The response of two competing nodes to patterns of three bits are shown in table 5.5, together with the summed response representing the activation of the virtual node formed by these two nodes. It can be seen that a virtual unit of two nonlinear nodes has begun to solve the problem, while a virtual unit of linear units fails. With two units there is still a need for synapses to participate in more than one cluster. This is only avoided when at least four units are used, in which case nonlinear nodes solve the 3-bit parity problem (table 5.6).

input pattern			parity	response of four nonlinear units				
0	0	0	0	0.0	0.0	0.0	0.0	*0.0*
0	0	1	1	0.0985	0.0	0.0	0.0	*0.0985*
0	1	0	1	0.0	0.0	0.1144	0.0	*0.1144*
0	1	1	0	0.0	0.0	0.0	0.0	*0.0*
1	0	0	1	0.0	0.1071	0.0	0.0	*0.1071*
1	0	1	0	0.0	0.0	0.0	0.0	*0.0*
1	1	0	0	0.0	0.0	0.0	0.0	*0.0*
1	1	1	1	0.0	0.0	0.0	0.0977	*0.0977*

Table 5.6: Learning the 3-bit parity problem with four nodes. The first three columns give the input patterns. The last five columns give the mean output responses of nodes after training for 16000 iterations. The fifth to eight columns are the responses of four nonlinear nodes, and the ninth column is the summed response of these four nodes ($\kappa = 0.00025$).

5.6.2 Contextual Biasing

Consider a problem in which there is more than one apical input. Such inputs can bias the formation of representations on the basal dendrite. Figure 5.37(a) shows data randomly distributed over the 2-dimensional plane. This data occupies two bands with x-coordinates ranging from 0.25 to 0.5 and from 0.5 to 0.75. This data is inseparable unless there is contextual information (such as plotting the data using different symbols; figure 5.37(b)). A 2-node network trained without any contextual information divides the data along the longest dimension, the y-axis (figure 5.37(c)). If the contextual information is supplied to the apical dendrites then competition between the nodes to represent the two apical inputs will affect the learning on the basal dendrites. The apical inputs were (1.0, 0.0) when the data was to the left, and (0.0, 1.0) when the data was to the right. The basal dendrites came to divide the input data into the two classes specified by the apical input (figure 5.37(d)). Hence, the contextual input to the apical dendrite modified the way in which the same input data was categorised. This will effect how input data



(a) (b) (c) (d)

Figure 5.37: Learning to classify input data. Input data is randomly distributed in two adjacent bands on the unit square of the 2-dimensional plane. Without contextual information the data is inseparable (a) and a 2-node network will divide the data along the longest axis (c). With contextual information the data is separable (b) and a 2-node network will divide the data into the correct classes (d). (c) and (d) Show the preferred inputs of the basal dendrites of the nodes in the network, projected onto the input space, after training for 4000 iterations.

is perceived. Such effects are known as categorical perception (Harnad, 1987b,a; Harnad et al., 1991).

Modifying learning using inputs supplied to the apical dendrites has been successfully used with both linear and nonlinear nodes, and with discrete and continuous input spaces.

5.7 Summary

The previous chapters set out some criteria for the type of neural representations that should be learnt. These criteria included forming place coded feature detectors, which are more locally coded (more abstract) than the input encoding. This chapter has shown that the algorithm presented in chapter 4 is capable of forming such representations in a robust and stable manner. However, to model development it is necessary to learn different levels of abstraction to control different behaviours. The following chapter demonstrates that by using the output of one network as the input to another that more abstract representations can be learnt. It also shows that such representations can be associated with motor actions in order to learn to control behaviour.

Chapter 6

INTER-REGIONAL DEVELOPMENT

This chapter considers the performance of simple assemblies of neural networks. The first half considers how the representational capacities of single model regions (investigated in the previous chapter) might be extended by using multiple regions, in which the input to one region is the output of another. The second half of the chapter considers the situation in which the activity of certain networks within an assembly cause actions which effect in the environment which may in turn have sensory effects. In this case the activity in one region can affect the input to another region either through direct synaptic contacts or via the environment. Such an assembly will be used to learn to control actions.

6.1 Cognitive Development

This section will consider the simple hierarchies of regions shown in figure 6.1. As with the assemblies used to test learning in single regions (figure 5.1) input data can come from single or multiple sources and learning may involve finding abstractions or invariances. Hence, each individual region within these assemblies is learning to represent input data in the same way as before. This section will, thus, simply demonstrate that the output of one region can be used to supply the input to another, and that more abstract representations may result.

6.1.1 Abstract Representation Learning

This section considers using an assembly like that shown in figure 6.1(a) to learn representations of more abstract categories in the input data. The bars problem (section 5.1) was modified so that a small proportion of the input data consisted of one of four predefined patterns (figure 6.2). These 'embedded'

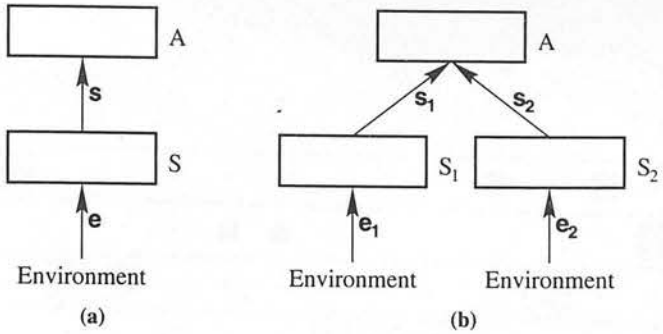


Figure 6.1: The assemblies used to test simple sensory abstraction and invariance.

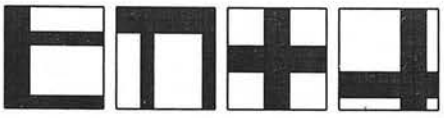
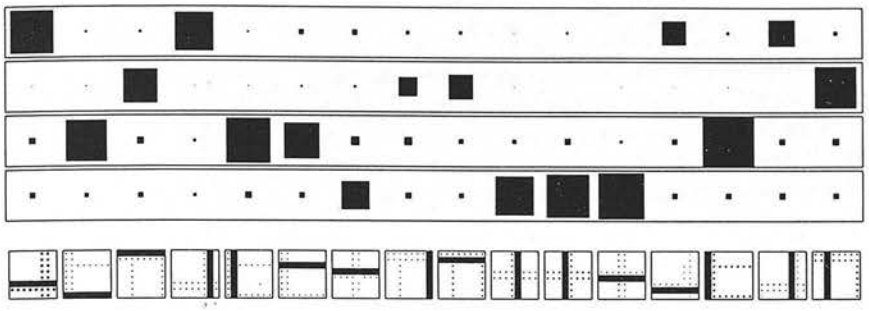


Figure 6.2: Additional patterns embedded into the bars data. These patterns were used in conjunction with the input patterns for the bars problem (figure 5.2(a)).

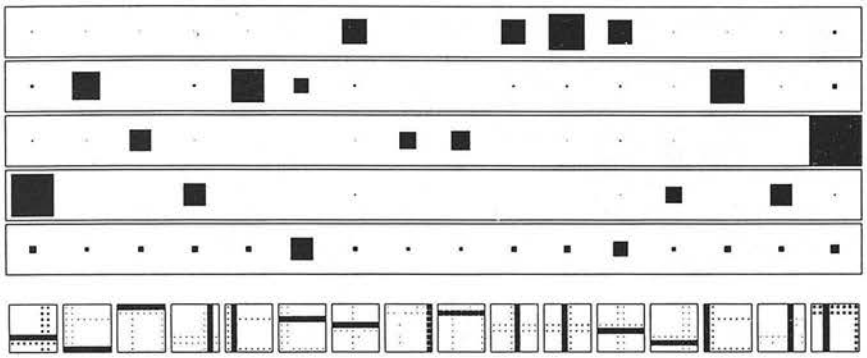
patterns each consisted of four bars and occurred at a significantly higher frequency (0.05) than they would by the chance coincidence of the four constituent bars in isolation (0.00005), while occurring less frequently than individual bars (0.125). When this data is supplied as input to the assembly, the network forming the lower region will learn a factorial representation of the individual bars, as in section 5.1. The network learns to represent individual bars, rather than the embedded patterns, due to the infrequency of the embedded patterns compared to the individual bars. However, a second neural network in the upper region, receiving this representation as input, can learn to identify the embedded patterns. If the upper network consists of fewer nodes than there are bars then individual nodes in the upper region will be forced to represent multiple bars. The presence of the embedded patterns will bias the selection of the set of bars represented by an upper node to include all those within the same embedded pattern.

Using 16 nodes in the network of the lower region a factorial coding of the individual bars was developed in 24% of 50 experiments with different randomly generated training sets. This result compares to the value of 35% found in section 5.1 for a 16-node network forming a suitable representation of the bars problem (without embedded patterns). Using 20 nodes in the lower region generated a factorial coding of the individual bars for 92% of 50 experiments, compared to 83% previously. With 20 nodes in the lower region the nodes which do not form a representation of individual bars have a preference to the embedded patterns. However, this preference is weaker than the preference of the other nodes to individual bars; the differentiation between the synaptic weights representing the preferred pattern and other inputs is lower, and the total sum of synaptic weights to the preferred pattern is smaller.

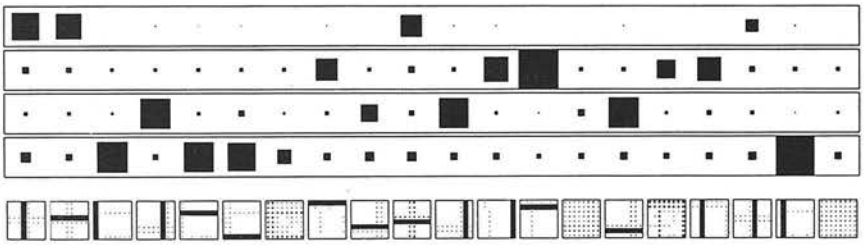
For experiments in which the lower region successfully generated a factorial representation of the



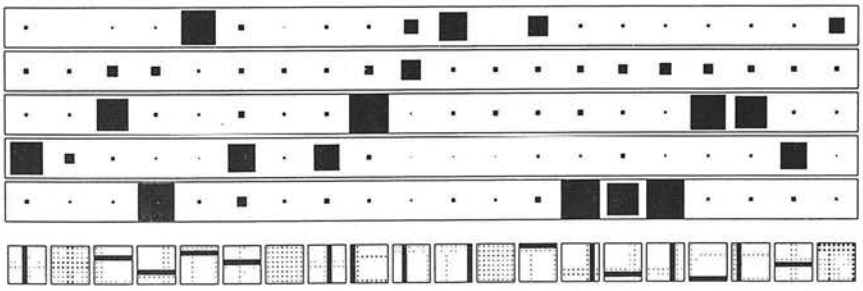
(a)



(b)



(c)



(d)

Figure 6.3: Learning to represent embedded patterns. A network forming the upper region received input from nodes in a network forming the lower region. The input to the lower region was from an 8 by 8 array of binary inputs receiving training data consisting of the bars data (as shown in figure 5.2(a)) together with embedded patterns (shown in figure 6.2) occurring significantly more frequently than by chance, but with lower frequency than for individual bars. The synaptic weights of each node are indicated by squares with size proportional to the strength of that weight. The weights of the nodes in the lower region, to the 8x8 input grid, are shown in the bottom rows. The weights of the nodes in the upper region are shown in the top four rows such that the weights from each node in the lower region are shown immediately above the representation of that node in the bottom row. Training consisted of 8000 pattern presentations. Results in which nodes in the upper region successfully represented all the embedded patterns are shown for varying numbers of nodes in each network. (a) A 16-node lower region and four-node upper region. (b) A 16-node lower region and five-node upper region. (c) A 20-node lower region and four-node upper region. (d) A 20-node lower region and five-node upper region. It can be seen that when there are excess nodes in the upper region the unused node has relatively weak synaptic weights, and when there are excess nodes in the lower region these tend not to be strongly connected to upper

Number of nodes in each region		Delay in start of learning in upper region	
lower region	upper region	0 iterations	2000 iterations
16	4	38%	94%
16	5	18%	92%
20	4	15%	40%
20	5	4%	32%

Table 6.1: The change in robustness with number of nodes in the network. 50 experiments with random variations in the training order were performed for each condition. The table shows the percentage of experiments, out of those in which a suitable representation of the bars was formed in the lower region, for which the nodes in the upper region formed suitable weights to represent the embedded patterns. Results are given for experiments performed with random variations in the training order and for a total of 8000 training cycles.

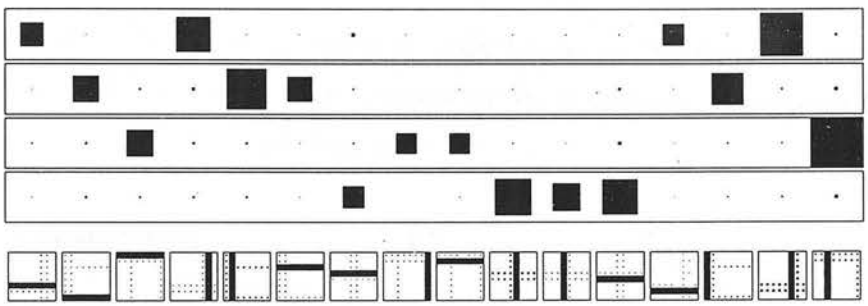


Figure 6.4: Learning to represent embedded patterns with delay. The figure should be interpreted in the same way as the results presented in figure 6.3. Identical experimental conditions to those used to generate the results in figure 6.3(a) were used except that the lower region was trained for 2000 cycles before the upper region was allowed to learn.

individual bars it was possible for the upper region to form a representation of the embedded patterns. The percentage of experiments for which the upper region successfully found suitable representations of the embedded patterns is shown in table 6.1, and examples of successful experiments are shown in figure 6.3. The ability of the upper region to detect the embedded patterns was greatly increased by allowing the lower region to learn for some time before allowing the upper region to begin learning. A delay in learning in the upper region sufficient to allow a suitable representation to form in the lower (*i.e.*, 2000 cycles) gave a large increase in the percentage of experiments for which a suitable representation of the embedded patterns was found (an example of a successful experiment is given in figure 6.4). While networks containing excess nodes in the lower region are more likely to find a suitable representation of the individual bars (necessary for the upper region to represent the embedded patterns), excess nodes in the lower region make it less likely that the upper region will represent the embedded patterns. Hence, it would seem necessary that in order to robustly discover abstract representations in the upper region it is necessary to allow a lower region, with excess nodes, to learn for some time first. Unused nodes in the lower region would then need to be eliminated before the upper region began learning. Unused nodes are easily identified and could be removed automatically, as discussed in section 5.1. The elimination

of cortical cells which fail to obtain effective stimulation is believed to result in the cell death that is observed during the critical period (section 3.2.3). Allowing regions to mature at different rates could be achieved by using a 'wave-of-plasticity' (Jacobs, 1999; Shrager and Johnson, 1996, section 3.2.3) travelling through the assembly, from the lowest regions towards the highest, such that lower regions mature first in order to provide useful and stable input data for higher regions.

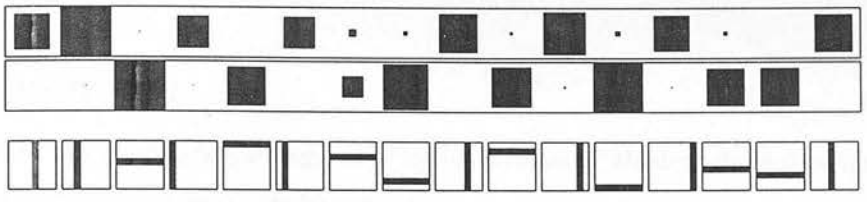
6.1.2 Invariant Representation Learning

Section 5.5 demonstrated that the sensitisation factor, λ_x, could be used to learn translation invariant representations given temporal correlations within the input data. The experiments in this section learn equivalent representations using multiple-region assemblies.

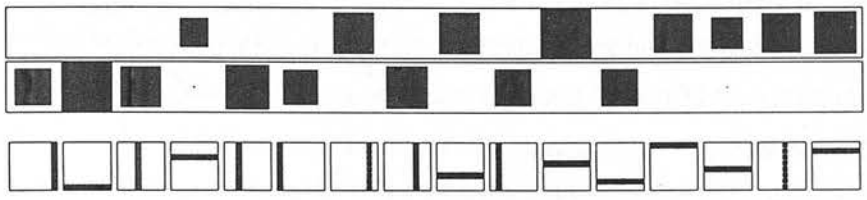
6.1.2.1 Horizontal and Vertical Bars

In section 5.5.1 invariant representations of horizontal and vertical bars were learnt using a single region of nonlinear nodes. The same invariant representation can be learnt using two regions of linear nodes, arranged in an assembly as shown in figure 6.1(a). If the lower region learns to represent individual bars (as in section 5.1) then each node in the upper region can learn to represent one orientation of bar by forming strong synaptic weights to all the nodes in the lower region which represent bars of that orientation.

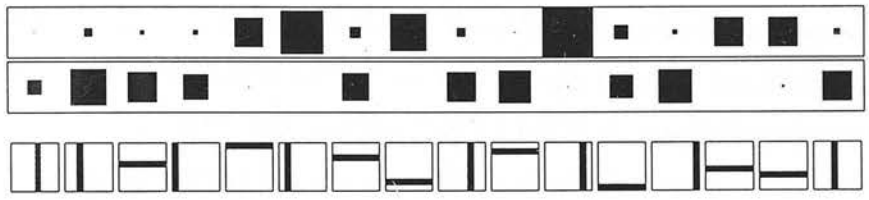
A network consisting of 16 nodes was used for the lower region and a network of two nodes was used for the upper region. Both networks used the standard parameter values except for $\lambda_x = 0.25$. The training data supplied to the lower region was the same as that used in section 5.5.1, consisting of short sequences of patterns containing single bars at a constant orientation (figure 5.35(a)). The nodes in the lower region found a suitable representation of the individual bars. Learning a suitable representation in the upper region, which correctly distinguished between horizontal and vertical bars, required learning in the upper region to be delayed (figure 6.5(a)). Without a delay or without using sensitisation the upper region nodes represented random sets of bars (figure 6.5(b and c)). The required delay was much longer than that found necessary in the previous experiment (section 6.1.1). While the lower region nodes found a suitable representation of the bars, in terms of weight decoding, within 1800 training cycles, learning in the lower region needed to take place for at least 4500 training cycles before learning in the upper region commenced. As before the delay time allows the lower region to find a stable representation of the bars before the upper region starts learning based on this representation. However, the required training time for the lower region is well in excess of the time required to find such a stable representation. This is due to the use of the sensitisation factor which causes the input to the lower region to appear to be more noisy than is the case without sensitisation. The output of the lower region is thus also more noisy and the activity of the winning node is less distinct from the activity



(a)



(b)



(c)

Figure 6.5: Learning to represent horizontal and vertical bars. A two-node network received input from a 16-node network which in turn received input from an 8 by 8 array of binary inputs. The synaptic weights of each node are indicated by squares with size proportional to the strength of that weight. The weights of the nodes in the lower region, to the 8x8 input grid, are shown in the bottom rows. The weights of the nodes in the upper region are shown in the top two rows such that the weights from each node in the lower region are shown immediately above the representation of that node in the bottom row. Training data consisted of one only of the 16 possible horizontal or vertical bars being active at any one time with short sequences of bars at the same orientation (as shown in figure 5.35(a)). (a) and (b) The lower network was trained for 5000 iterations before the upper network was allowed to learn for a further 3000 iterations. (a) With sensitisation ($\lambda_x = 0.25$) one node in the upper region represents all horizontal bars while the other represents all vertical bars. (b) Without sensitisation ($\lambda_x = 1.0$) the nodes in the upper region represent an arbitrary selection of bars. (c) Both networks were allowed to learn from the start, for 8000 iterations. Even with sensitisation ($\lambda_x = 0.25$) the nodes in the upper region fail to distinguish horizontal and vertical bars.

of all other nodes. Continued training is necessary for the activity of the winning node, which represents the presence of a bar in the input, to become sufficiently distinct from the activities of the other nodes, and hence to provide useful data for input to the upper region. As well as an increased delay period stronger sensitisation (a lower value of λ_x) was also found necessary. In all previous experiments using sensitisation, $\lambda_x = 0.5$ has been used. However, it was found that in this case the use of $\lambda_x = 0.5$ made classifying horizontal and vertical lines unreliable (even with a delay in learning in the upper region only 12% of 50 experiments with random variations in the training order generated correct solutions). Sensitisation causes the inputs to the lower region to appear noisy, resulting in noisy output from the lower region. Sensitisation of the inputs to the upper region increases the apparent noise on the output of

the lower region. Rather counter intuitively, stronger sensitisation is required; although this will result in more apparent noise it will also increase the tendency to learn temporal correlations. A better solution might be to allow sensitisation to affect learning only, and not to allow it to affect activation values. This would prevent sensitisation from increasing the noise at each stage in the hierarchy. When using a delay of 5000 iterations before the upper network was allowed to learn and $\lambda_x = 0.25$ correct solutions were found in 52% of 50 experiments with random variations in the training order. A typical successful result is shown in figure 6.5(a). (44% of 50 experiments with random variations in the training order were successful with $\lambda_x = 0.375$.) When the upper region does learn a suitable representation then one of the nodes will respond to an input containing a horizontal bar while the other node will respond to an input containing a vertical bar. These nodes will respond more strongly to several bars of the same orientation presented simultaneously, or to a single bar observed for a temporally extended sequence in several positions. Inputs containing multiple bars with both orientations will cause both upper nodes to be active.

6.1.2.2 Corresponding Bars

In section 5.5.2 a single region learnt translation invariant representations of individual bars across two input streams. The assembly shown in figure 6.1(b) can learn an equivalent representation. In this case the two data streams are input to separate, lower, regions. These regions learn to represent each individual bar in each input stream. The output of these lower regions then provides the input to an upper region which can be used to learn to represent corresponding bars within the two streams. Such an assembly was supplied with training data identical to that used in section 5.5.2 in which the bars data was supplied to alternating data streams in a sequence of random length (between 0 and 10 presentations). For each presentation of the input pattern to one data stream the other stream received either a blank input or another random bars pattern. If training in the upper region was not delayed then only some upper nodes (more than half) became responsive to corresponding bars in the two streams, however, some of these correlations were weakly represented. Much improved results were obtained if training in the upper region was delayed (*i.e.*, for 4000 cycles) until both the lower regions had formed suitable representations of the bars (figure 6.6). Only a few experiments were performed and figure 6.6 represents a typical successful result. The required delay time is similar to that required above as sensitisation causes the input to the upper region to appear much more noisy. However, unlike the above experiment $\lambda_x = 0.5$ was found to be more affective than $\lambda_x = 0.25$. This is probably due to there being multiple bars in the input patterns so that a lower value of λ_x would cause more apparent noise to the upper region than was the case previously where patterns contained only single active bars.

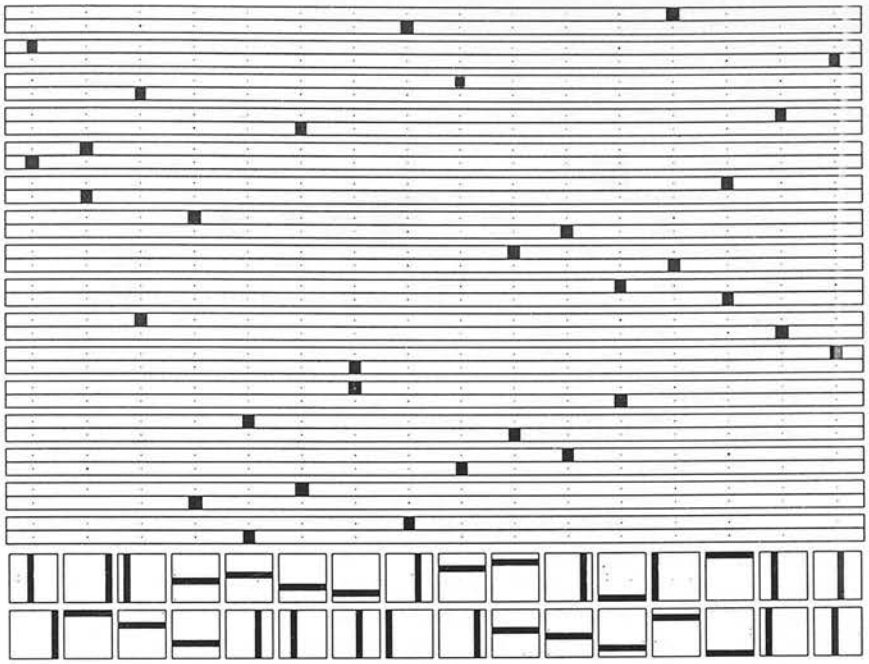


Figure 6.6: Learning to represent a specific bar invariant to translation. Input patterns containing bars on an 8 by 8 grid were supplied to each of two input streams converging on two networks of 16 nodes. The same pattern was presented to one data stream and then the other repeatedly for sequences of variable length. The lower networks were trained for 4000 iterations before the upper network was allowed to learn for a further 4000 iterations. Sensitisation was used ($\lambda_x = 0.5$). The bottom two rows show the strength of synaptic weights for the two lower regions. Both lower regions supplied input to a single upper region of 16 nodes. The upper rows show the weights of each of the 16 nodes in the upper region, these weights are shown using two rows per node, each row corresponding to inputs from one of the two lower regions.

6.1.2.3 Generalised Invariant Representations

In all the previous examples invariance is learnt by presenting the pattern at all locations in which it is to be recognised. This requirement is true of most other models of invariant pattern recognition. An exception are neural networks which are structured specifically to learn translation invariance (*e.g.*, by using shifter circuits (Olshausen et al., 1993) or weight sharing (Fukushima, 1980)). For example, weight sharing neural networks update corresponding weights in separate receptive fields of the same neural layer to be the same: presenting a pattern at one position causes the synaptic weights to all other positions to update in the same way. Hence, translation invariance is built in to the neural architecture rather than being learnt from experience, and this mechanism is biologically implausible (O'Reilly and McClelland, 1992). Neural networks that do not have built in translation invariance have generally failed to demonstrate generalisation of pattern recognition from one part of the input array to another without being trained with patterns presented at all locations. Hence, they fail to model learning of invariant recognition in the visual system where after training with a single view of a novel, complex, object cells in the inferotemporal cortex show some degree of invariance centred around the view that has been seen

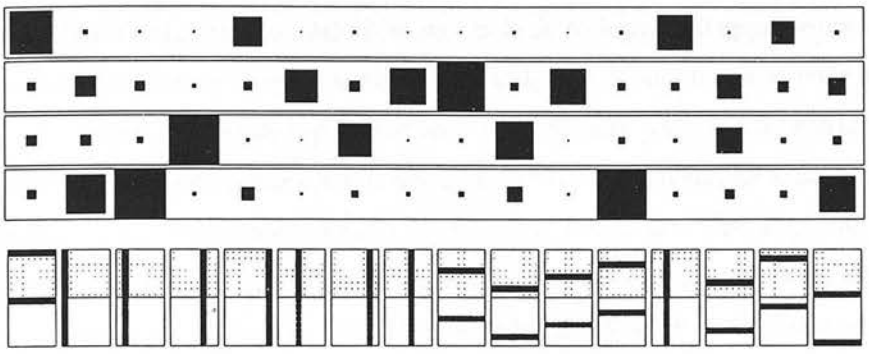


Figure 6.7: Learning an invariant representation of an embedded pattern. A 16-node network received inputs from two data streams. The bottom row shows synaptic weights of nodes in that network with weights to input streams shown one above the other. This network has learnt to represent corresponding bars in each stream. An upper region began training after the lower region had been trained for 4000 iterations, and training continued for a further 4000 iterations ($\lambda_x = 0.5$). The weights of the four nodes in the upper region are shown in the top four rows. The upper nodes have learnt to represent patterns which were presented to one input stream but not the other.

(Riesenhuber and Poggio, 1998b,a).

However, such generalisation of invariance may develop over many processing stages (Wallis and Bülthoff, 1999). It was shown, in section 5.5.2 and immediately above, that a region receiving input from two correlated data streams can learn an invariant representation across those streams. Consider such a network providing input to another region which learnt to represent embedded patterns within the input data. In this case embedded patterns presented to one data stream could be learnt by the upper region, and, due to the invariant representation in the lower region, such patterns could then be recognised when presented to the other data stream, even if never presented to that stream before. Hence, an invariant representation of the embedded patterns would be learnt and because the invariance of the subfeatures has already been learnt it is not necessary to see multiple views of the embedded pattern to be able to recognise it in multiple locations.

A network was trained with data in which one pattern, containing a single active bar, was presented alternately to each data stream for a random sequence length (between 0 and 10 iterations). The embedded patterns shown in figure 6.2 were also presented, with probability 0.05 each, to one input stream but not to the other. Nodes in the lower region learnt to represent pairs of corresponding bars in the two input streams. It was found that the upper region could learn to represent the embedded patterns if training of the upper region was delayed until the representation had formed in the lower region. Figure 6.7 shows the synaptic weights of such an assembly in which the upper region was trained for 4000 iteration after the lower region had been trained for 4000 iterations. A sensitisation factor of $\lambda_x = 0.5$ was used. Although sensitisation is required to learn the receptive fields of the nodes in the lower region, it is not required to for learning in the upper region. Hence, a sensitisation factor is required that is sufficiently strong for learning the lower region, but is sufficiently weak not to generate too much noise in the input

to the upper region.

It can be seen from figure 6.7 that the upper region has formed a representation of the embedded patterns. Since the lower region has very similar receptive fields from both input streams the same pattern in either stream will cause similar activations in both the lower and upper regions. In particularly, the same embedded pattern supplied to either input stream will cause the same node in the upper region to respond. Hence, an invariant representation of the embedded patterns has been learnt. Complementary subfeatures of the same embedded pattern applied to both input steams simultaneously will cause the same activation as the complete pattern applied to just one stream. Hence, as with all models that learn to recognise a pattern invariant to location, information about the pattern's location is lost. Such models (like the ventral visual pathway) process information about 'what' a visual stimulus is, and would seem to require a complementary mechanism for recording 'where' the stimulus is.

6.2 Behavioural Development

In this section neural assemblies that connect sensory input to motor outputs will be considered. Although neural networks will be described as sensory and motor regions it is not suggested that they correspond to specific cortical regions. In addition, it will be useful to describe these assemblies as learning specific sensory-motor tasks, such as hand-eye coordination, but they should not be interpreted as a model of animal behaviour.

The following experiments demonstrate a developmental progression in behavioural complexity by going through various stages. Assemblies of neural networks will be considered which can learn a simple reflexive skill, learn how to control a simple skill and learn how to combine simple skills to generate more complex behaviour. However, for this to be possible it will be necessary to provide some simple innate abilities as a prerequisite for learning even the simplest skill. (Previous experiments in this thesis have also relied on innate abilities; *e.g.*, the ability to see is a prerequisite for classifying visual sensory inputs.) To illustrate the type of behaviour that will be learnt consider a simple, two-dimensional, agent which has an eye, an arm, and a waist (figure 6.8). Such an agent could rotate its eye to fixate on visual targets, move its arm to reach for objects, and lean its whole body backwards and forwards. An example of the behavioural progression described above in such an agent would be the following.

1. Innate skill; the ability to fixate on visual targets.

2. Simple skill; the ability to reach to the point of fixation (simple hand-eye coordination).

3. Complex skill; the ability to control reaching to reach for objects and not to reach for free-space.

4. Coordinated skill; the ability to lean forward to reach for an object initially beyond arm's length.

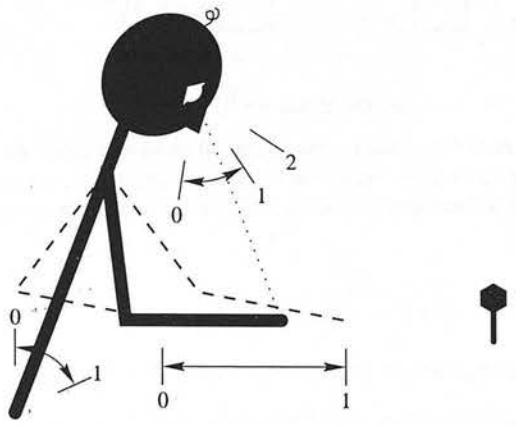


Figure 6.8: A schematic of a simple agent for use in behavioural development. The agent can visually fixate points resulting in a direction of gaze, it can move its arm so that its hand moves along a line, and it can lean so that the whole body is translated. All movements are in the 2-dimensional plane.

By fixating on its own hand the correspondence between arm position and gaze direction could be learnt. Such a mapping would allow the agent to move its hand to the current point of eye fixation and hence to reach for visual targets. However, such a simple mapping would be evoked even when the agent was gazing at free-space and hence a gating mechanism is required to allow the agent to learn to reach only when it sees an object. The agent would also fixate on objects beyond arm's length but fail to be able to reach them. However, if the agent happened to lean forward far enough to bring an object into reach the hand-eye coordination skill would cause the hand to touch the object and this could provide a cue for learning to lean when fixating distant objects and hence to combine these behaviours to allow the agent to 'purposefully' move in order to reach objects.

6.2.1 Simple Skills

6.2.1.1 Reflexive Behaviour

A prerequisite for the performance of a skill is knowing the relationship between actions and their effects. For a simple skill this may be achieved by finding the mapping from the sensor domain to the motor domain. When these domains are represented by neural networks (figure 6.9) it is a case of finding appropriate synaptic weights to connect the motor region to the outputs from the sensor region. It is thus necessary to learn appropriate receptive fields for the nodes in the motor region, in the same way that the nodes in the sensor region must also learn appropriate receptive fields to represent sensory input. Similar neural assemblies have previously been used to learn simple mappings such as hand-eye coordination (Churchland, 1990, 1986; Mel, 1990a; Salinas and Abbott, 1995; Balkenius, 1995); and methods of finding the mapping in such an assembly have previously been presented as models of the cerebral cortex (Churchland, 1990; Salinas and Abbott, 1995).

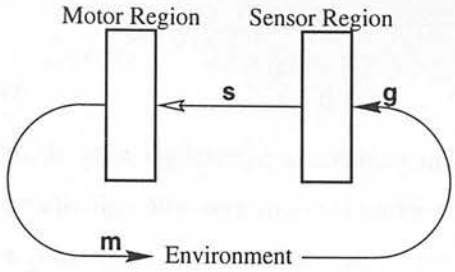


Figure 6.9: The assembly used to learn simple sensory-motor mappings consists of two regions: a motor region generating the motor output and a sensor region receiving inputs in response to the motor actions. These regions are joined by connections which generate the required mapping.

To learn the transformation between sensor and motor space requires training data covering the range of possible actions, and in which sensor input corresponds to current motor output. Thus, most algorithms (*e.g.*, Churchland, 1990, 1986; Salinas and Abbott, 1995; Mel, 1990a; Balkenius, 1995; Schulten and Zeller, 1996; Cohen et al., 1997; Gaudiano and Grossberg, 1991; Coiton et al., 1991) go through a distinct training phase during which the output of the motor region is determined by a random number generator, and the inputs to the sensor region are set to corresponding values from this training set (as if the sensor data was generated by the random motor actions). There is thus a distinct, externally imposed, training regime during which the motor region implements a different algorithm to that which is implemented when the resulting mapping is used to control behaviour. Such a separation between training and control is not observed in the development of animals (Metta et al., 1997). In addition, such a scheme is biologically implausible as it is unlikely that the motor cortex changes the algorithm it implements (requiring neurons in the brain to switch between behaviours), or that motor actions are under 'external' control during development, or that such a training regime is intermittently used during growth to maintain sensory-motor alignment. A distinct training phase would also be problematic when learning more complicated behaviours, using an assembly with more regions, since a number of distinct and coordinated training phases might be required to develop associations between many cortical areas.

Using the neural network algorithm presented in chapter 4 avoids the use of any separate training phase; the mapping is learnt at the same time as the motor region generates (initially inaccurate) outputs in response to the sensory input. Hence, the algorithm implemented by each region remains the same throughout time. Habituation and noise in the selection of the winning node are important for learning useful representations of input data in individual regions; these mechanisms are also important for producing the training data to learn the mappings between regions. Initially, when the connections to the motor region are weak, the output of the motor region will be almost entirely random (the output will also be weak, but it is assumed that even weak activations have motor effects). As the connections become stronger there is a tendency for the current sensor input to re-activate the previous motor output, and hence produce the same sensor and motor effects continuously. However, habituation prevents this

from occurring for more than a few iterations, allowing learning to continue and ensuring coverage of the range of possible actions[1]. Hence, the same algorithm performs exploration (to calibrate the control system) and exploitation (to perform the control task). Habituation will ensure that exploration continues even after the task has been learnt successfully. Similarly, the 'clean-up effect' (Michie, 1995) demonstrates that adult humans make sub-optimal, exploratory, movements when performing a task in which they have demonstrated knowledge of the optimal behaviour in the past.

Learning must be under-pinned by an innate facility that ensures the sensor input corresponds to the current motor output. For hand-eye coordination to learn the correspondence between arm position and gaze direction would require an innate ability to frequently fixate the hand to provide the necessary training data. Fixation is an innate ability, subcortically controlled during early infancy in humans (Johnson, 1997, 1993; Atkinson and Braddick, 1989). Fixation will get attracted to foci of attention other than the moving hand, however, other constraints such as the infant's limited depth of field (Turkewitz and Kenny, 1993) and limited self-motion may help to limit the number of potential fixation targets and hence ensure that the hand is a sufficiently common target to provide the necessary training data. Even so, the gaze direction will not always correspond to the hand position and any learning algorithm ought to be able to cope with a significant number of mismatches.

As a simple experiment in learning sensory-motor correspondences the assembly in figure 6.9 was used. Two networks of nodes, a sensor and a motor region, were connected such that the output from the sensor region formed the input to the motor region. Each node in the motor region had a predefined preferred action (but not a predefined preferred input). The preferred actions of nodes were predefined to evenly cover the range of values 0 to 1 (*i.e.*, the motor region consisted of a one dimensional array of nodes, 1 to n, node 1 was assigned a preferred output of 0, node 2 was assigned a preferred output of $\frac{1}{n-1}$, and so on with each node i assigned a preferred output of $\frac{i-1}{n-1}$). The output activity of the motor region was decoded, as a coarse code, in terms of these predefined motor effects (*i.e.*, by taking the normalised sum of the activations weighted by the preferred output of each node: $action = \frac{\sum_{i=1}^{n} y_i Act_i}{\sum_{i=1}^{n} y_i}$, where y_i is the output activation of node i, and Act_i is the predefined preferred action assigned to node i). The decoded motor output was a value between 0 and 1 and constituted the action performed (*e.g.*, the arm position moved to). The input to the sensor region was also a value between 0 and 1 and constituted the sensor input (*e.g.*, the gaze direction). This sensor data was provided to the inputs of the sensor region as a coarse coded representation. The input to the sensor region was either the motor action (in which

[1] In this case the training data is oscillatory over the range of possible actions, rather than random. If sub-actions were represented by several, distinct, motor regions connected directly to the sensor region then synchronous oscillations might occur across the range of each action, leading to only a subset of the combination of actions being expressed. Hence, coverage of all motor actions is only ensured if the motor region represents all combinations of actions. It is thus necessary that the sensory-motor mapping is learnt at an appropriate level of abstraction in which all sub-actions are represented by a single network. This higher-level motor network could control the behaviour of lower-level networks expressing sub-actions.

case the value encoded in the input was equal to the value of the decoded motor output)[2] or a random value (also in the range 0 to 1). The probability of the sensory input corresponding to the current motor output was p_f which, for a hand-eye coordination task, represents the reliability of the innate fixation ability. Hence, for a fraction of p_f occasions, corresponding sensor and motor values are represented on the inputs and the outputs of the assembly. To ensure that the motor outputs are associated with their sensory consequences the synapses which form the inputs to the motor region were modified before the next motor output was calculated (*i.e.*, the efferent learning scheme was used; section 4.2.3).

Figure 6.10 shows a typical development of a simple sensory-motor mapping when two one-dimensional arrays of nodes were used to represent the sensor and motor regions, and with $p_f = 1.0$. Maps of preferred inputs have been generated by decoding the synaptic weights of the nodes. As well as showing these values in a simple map they are shown projected onto a pictorial representation of the connectivity of the two regions. Corresponding sensor inputs and motor outputs are shown at equal height. It can thus be seen that motor nodes are most strongly activated by those sensor nodes which represent the region of the sensor space in which the motor node generates data. Both networks form smooth, well ordered, topological maps in which there is monotonic progression in the preferred input of each node across the network. (As with the maps of 2-dimensional data generated in section 5.3 the measurement of the preferred input at the edge of the map is biased towards the centre.) Once formed the mapping is stable. The mapping is initially inaccurate, but develops to be more accurate during training. Since the output of the motor region should reflect the input to the sensor region, which in turn corresponds to the output of the motor region at the previous iteration, the accuracy of the mapping can be measured by taking the difference between consecutive motor outputs, as shown in figure 6.11(a). The training data is not random, but is generated by the output of the motor region. Initially, the synaptic weights, and hence the strength of lateral inhibition, is small and each node has a similar output activity; the decoded output is thus similar at each iteration (it is approximately the mean of the preferred actions), and hence the mapping accuracy is initially very good. As weights develop, and lateral inhibition increases, the range of output values generated also increases, and hence so does the error. However, this increased range of output values also provides training data and as the correct connections to implement the mapping are learnt so the error reduces. The profile of mapping accuracy in figure 6.11(a) is thus due to the training method rather than true accuracy in the connectivity between the regions. Measuring the output error to random input values (figure 6.11(b)) shows only a decrease in mapping error over time. Since the habituation mechanism deliberately introduces inaccuracy into the mapping to provide training data the error value can never reduce to zero. The mapping error is fairly evenly distributed across the range of sensor inputs but is typically slightly higher towards the edges of the map (figure 6.12).

[2] There is thus a direct correspondence between the value of the motor output and the value of the corresponding sensor input. A nonlinear correspondence, calculated as if the gaze direction needed to rotate to fixate the motor output, was also used. This gave very similar results, however, for presentation purposes only results for the linear correspondence are shown.

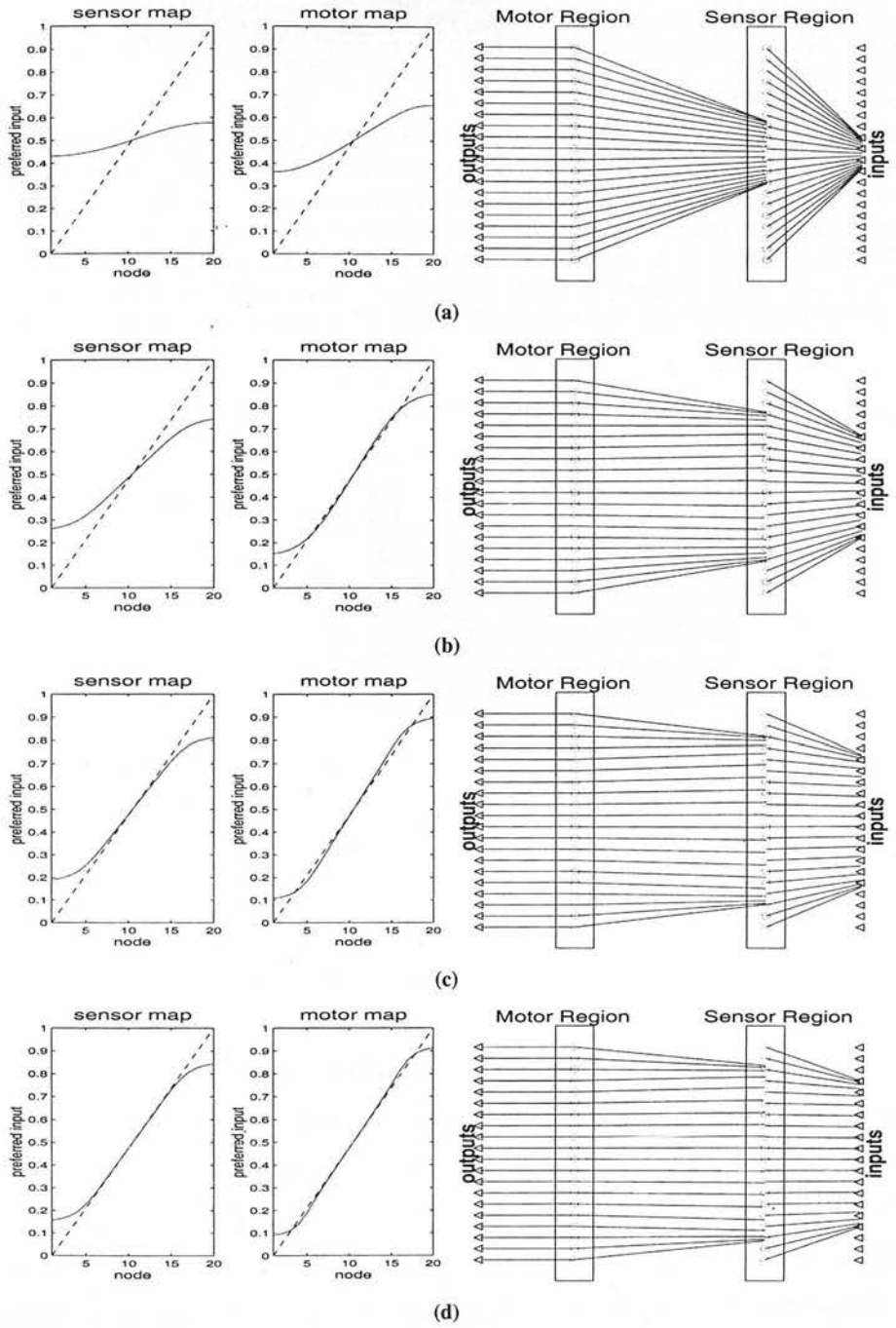
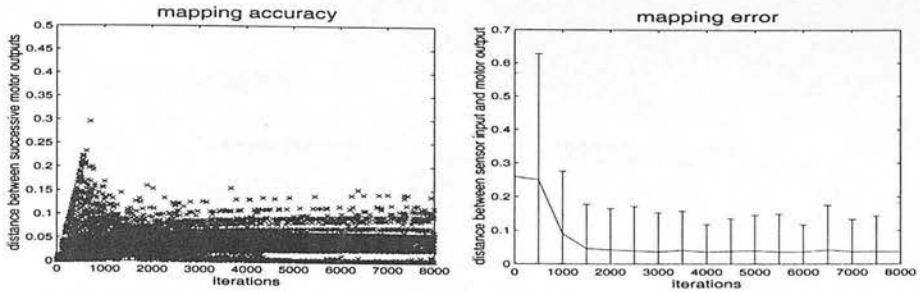


Figure 6.10: The development of sensor and motor maps. Motor and sensor networks contained 20 nodes each, and were training with 100% accurate fixation. The first two columns show the preferred input of each node. The third column shows the location of the preferred input, measured by decoding the synaptic weights, for each node and hence the mapping that is represented. Results are shown after training with (a) 1000, (b) 2000, (c) 4000, and (d) 8000 input patterns.



(a) (b)

Figure 6.11: The accuracy of a simple sensory-motor mapping. The variation over time of the error between the target position specified by the sensor input and the subsequent target position generated by the motor output, for motor and sensor networks each containing 20 nodes (as shown in figure 6.10). (a) The accuracy is measured using the training data generated by the motor region. The banding of the points is due to the algorithm selecting a single winning node at each iteration which generates quantisation error. (b) The error is measured by testing with randomly generated sensor input. Each point on this graph shows the average error over 500 tests and the error bars show the maximum and minimum errors during these tests.

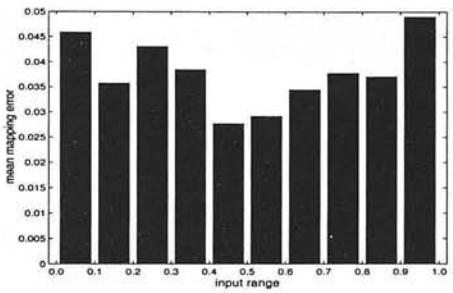


Figure 6.12: The distribution of mapping error across the input space. The mapping error, for motor and sensor networks each containing 20 nodes (as shown in figure 6.10), was measured for different ranges of input value after training for 4000 iterations.

Learning was robust to random variations in the training order as well as to random parameter values with 98% and 100% (respectively) of 54 experiments generating a suitable mapping in terms of the motor outputs being connected to corresponding sensor inputs (the mapping not being 'twisted') and being smooth. Since the orientation of each mapping is not predefined acceptable maps can also be as shown in figure 6.13(c). Unacceptable mappings would have one or other of the sets of weights twisted, or have non-smooth mappings. Figure 6.13 shows the synaptic weights learnt after 4000 iterations with different numbers of nodes in each region. All results have been generated using identical learning algorithms (including the same, standard, values for the parameters) in both regions. It is clear that the algorithm is robust to changes in the network sizes, and that very similar patterns of receptive fields are generated in all cases. The residual error is also similar in all cases since the width of the encoding of the input to the sensor region remains the same. Reducing this width can improve the accuracy of the mapping (figure 6.14) since it makes different input values more distinct and hence improves

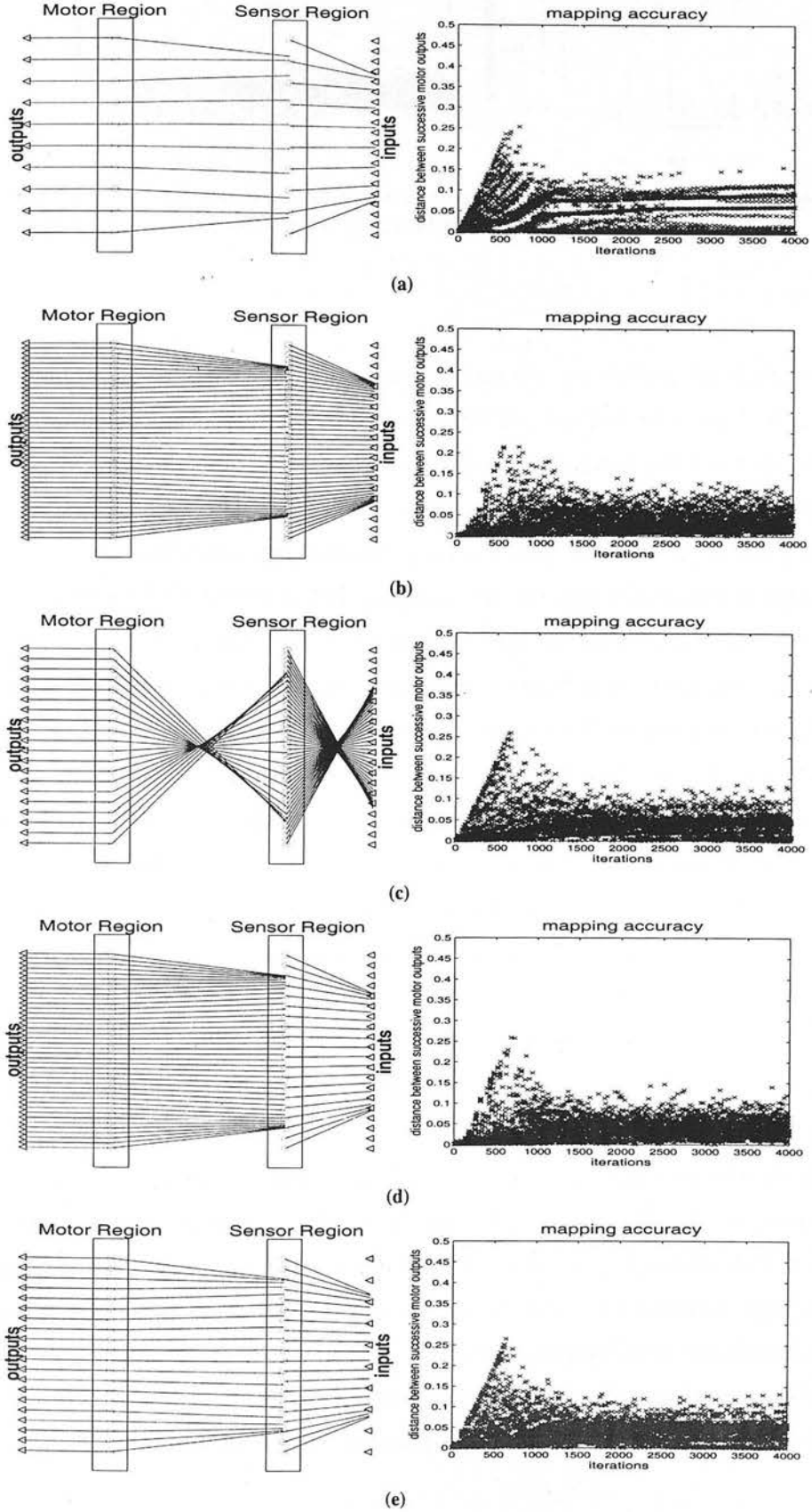
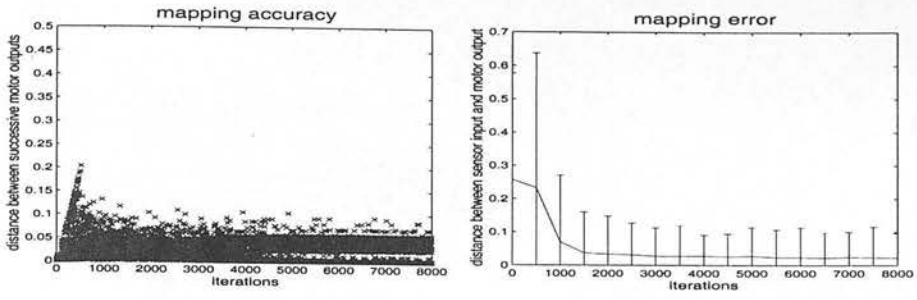


Figure 6.13: Sensory-motor mapping with varying numbers of nodes and numbers of inputs. The first column shows the location of the preferred input, measured by decoding the synaptic weights, for each node and hence the mapping that is represented (after 4000 iterations). The second column shows the variation over time of the error between the target position specified by the sensor input and the subsequent target position generated by the motor output. Results are shown for varying numbers of nodes in each network. (a) Both networks contain 10 nodes. (b) Both networks contain 40 nodes. (c) The sensor network contains 40 nodes and the motor network 20 nodes. (d) The sensor network contains 20



(a) (b)

Figure 6.14: The accuracy of a simple sensory-motor mapping. As figure 6.11 except the width of the coding for the input data (σ_D) is half the value used for figure 6.11.

the discrimination of the output of the sensor region (the same effect was discussed in section 5.3 for the map of the 2-dimensional plane, see figure 5.21). Narrowing the width of the input encoding will also sharpen the selectivity of the nodes in the sensor region leading to a reduced population of active nodes and hence (as for the sensor region) improved discrimination in the output of the motor region. However, all subsequent results are generated using the same, larger, width of input encoding as used in figure 6.10. With very few nodes forming each map quantisation of the error can be seen (figure 6.13(a)). This is due to the neural network algorithm selecting a single winning node at each iteration. The outputs of the motor region, thus, cluster around the preferred actions of individual nodes. A dynamic implementation of the algorithm (*i.e.*, equation 4.13) would allow smoother interpolation between nodes and hence provide better resolution. A coarse code, properly implemented, would be efficient for generating coordinate transformations since accuracy would not be limited by the number of nodes in the network; it could allow better resolution for a fixed number of nodes, or equivalently, require fewer nodes to represent data with a given accuracy.

Two sources of randomness can be introduced into the innate fixation ability: unreliability of fixating on the hand; and inaccuracy in the direction of fixation. The first case corresponds to reducing p_f (so far all experiments have used $p_f = 1.0$, so that the input to the sensor region corresponds to the previous output of the motor region at all times). In experiments with $p_f < 1.0$ the formation of a suitable mapping was robust; performing 54 experiments with random variations in the training order, for each value of p_f =0.75, 0.5 and 0.25, suitable mappings were generated for at least 98% of the experiments for each p_f. The noise that fixation on other targets induces is evenly distributed across all combinations of arm position and gaze direction and so does not influence the alignment of the mapping (when the activation of the sensor region represents a random gaze direction the weights between the active motor nodes and the active sensor nodes will increase, but such increases are evenly distributed across all possible inputs to the motor region and so do not affect the direction of the preferred inputs). Hence, actions are only associated with consistent consequences due to the use of efferent learning and the algorithm

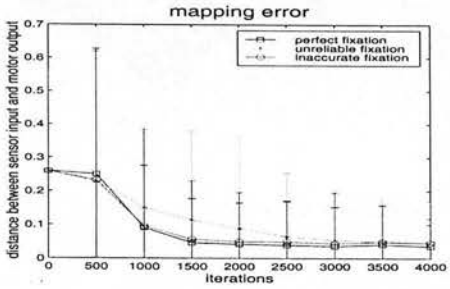


Figure 6.15: The accuracy of a simple sensory-motor mapping with randomness in fixation. The variation over time of the error between the target position specified by the sensor input and the subsequent target position generated by the motor output, for motor and sensor networks each containing 20 nodes. The error is measured by testing with randomly generated sensor input. Results are shown for training performed with perfect fixation, for training when fixation is only reliable 25% of the time (and input is random on the other 75% of occasions), and for training when fixation is inaccurate by up to 25% of the full range of input values.

is robust even if the agent spends most of its time looking at other targets than the hand. Decreasing p_f results in slower learning as can be seen from figure 6.15 which shows the change in accuracy over time when fixation reliability is only 25% (*i.e.*, $p_f = 0.25$). This is to be expected as reducing the number of accurate fixations reduces the number of training examples retarding the differentiation of the weights to the motor region.

The second source of randomness in the fixation process is due to fixation being inaccurate, in which case the input to the sensor region corresponded to the previous motor output deflected by a random distance uniformly distributed in the range $\pm a_f$ (since sensor values ranged from 0 to 1 the value of a_f corresponds to a proportion of the full range of possible sensor values). Under these conditions the formation of the mapping was still robust with at least 98% of 54 experiments, using random variations in the training order, finding a suitable mapping for all values of $\pm a_f$ used (0.05, 0.1, 0.25, and 0.5). Learning was only slightly slower than for the situation where there was no randomness in the sensor inputs but the final accuracy of the mapping was slightly reduced. Figure 6.15 shows the change in accuracy over time for a typical experiment in which there was a $\pm 25\%$ inaccuracy in the fixation (*i.e.*, $a_f = 0.25$).

Initialising the synaptic weights to be random values uniformly selected from the range 0 to v caused the mapping to be unsmooth, and for the mean mapping error to double, after training for 4000 cycles, when $v = 0.01$. Higher values of v prevented any topological organisation of the mapping from being formed and caused very poor accuracy. The mapping was more robust to corruption of the weights after training. Adding random values uniformly selected from the range $-v$ to $+v$ to the synaptic weights of nodes in both regions after 4000 training cycles caused the average mapping error to increase with increasing v. The mapping error was doubled for $v = 0.01$, however since the average synaptic weight in each region was less than this value of v ($\bar{q} = 0.0081$ in the sensor network and

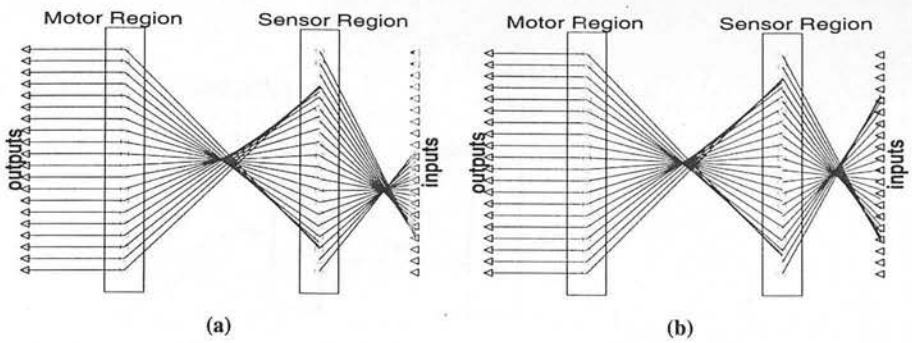


Figure 6.16: Plasticity of a simple sensory-motor mapping. Sensor input was generated such that the relationship between the motor output and the corresponding sensor input changed smoothly for the first 4000 iterations, after which it remained constant and the same as that used in all previous experiments. During the first 4000 iterations the sensor input was calculated to be $\frac{I}{4000}$ times its normal value. The coarse coded representation of the input is thus, initially, shifted with respect to its normal position towards the lower half of the input array. (a) Results after 4000 iterations. (b) Results after a further 4000 iterations.

$\bar{q} = 0.0068$ in the motor network) this represented a significant degree of corruption. Since the output of a region is due to the action of a population of nodes, errors in the activation of individual nodes are averaged out. Hence, coarse coding is robust to noise in individual neuron activations and it is also robust to node failure.

The mapping proved to be plastic to changes in the relationship between motor output and sensor input. In the case of hand-eye coordination a change in the correspondence between hand position and gaze direction might result from growth of the physical system and would thus be a systematic change over an extended time period. A network was trained for 4000 cycles during which the relationship between gaze direction and hand position changed continuously, and smoothly, until at 4000 cycles the correspondence was as used in the previous experiments (*i.e.*, the sensor region input was made equal to the motor region output but scaled by a factor of $\frac{\min\{I,4000\}}{4000}$, hence motor outputs in the range 0 to 1 were mapped onto sensor inputs in the range 0 to $\frac{\min\{I,4000\}}{4000}$, and the coarse coded representation of the input data was thus shifted across the input array). Figure 6.16 shows the mapping produced after 4000 cycles of training and after a further 8000 cycles of training with a constant correspondence (similar results are found for sudden changes in the relationship between sensor and motor spaces). It can be seen that the resulting mapping is comparable to that generated with a fixed correspondence between sensor and motor spaces (figure 6.10). This is achieved without specific re-training but as a consequence of the normal operation of the algorithm.

This section has shown how the relationship between two networks of artificial neurons, representing different cortical regions, can be learnt. Given a region of motor neurons whose activity has sensory consequences, and a network of sensor neurons which receive sensory input corresponding to the motor output, it is possible to learn the mapping between these domains. The algorithm enables each neural

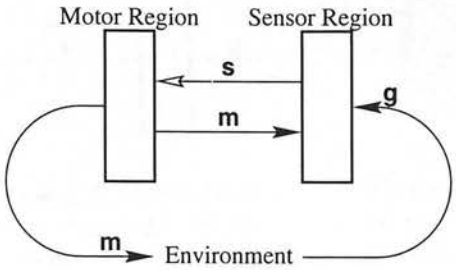


Figure 6.17: An infeasible assembly for learning motor-sensory mappings.

network to self-organise into a representation of its domain at the same time as the relationship between these domains is found. The architecture proposed here is very similar to that used by Salinas and Abbott (1995) in that it learns the mapping between two coarse coded neural networks. However, unlike that model it does not require that the receptive fields of the nodes in the sensor region are predefined (instead both regions are modelled in the same way) nor does it use a separate, externally generated, training phase (instead the model remains the same throughout time).

At the same time that the mapping from sensor space to motor space is being formed reciprocal connections could be used to learn the mapping from motor space to sensor space and hence to learn to predict the visual position of the hand. However, it is not possible, with the current algorithm, to have a single network receiving signals representing both current and expected sensory inputs (nor current sensory inputs and required motor actions), since these sources would interfere with each other, and the output of the region would need to represent both inputs. Hence, the assembly in figure 6.17 has not been used, however, section 7.2.2 suggests how the model could be extended to achieve this behaviour.

6.2.2 Complex Skills

6.2.2.1 Gated Reflexive Behaviour

By connecting two neural networks together a mapping from a sensor network to a motor network has been learnt. However, such a simple mapping can only model simple reactive behaviour. One way in which more complex behaviour may be generated is by allowing nonlinear interactions between domains; for instance, so that reactive behaviours can be gated by other sensory inputs. For this purpose the model of a nonlinear neuron, described in section 4.4.2, will be used in this section.

Using the regional assembly described in the previous section an agent would always follow its gaze with its hand, regardless of whether it was gazing at an object or at free-space. The addition of visual information can allow the agent to learn to reach only for objects. Figure 6.18 shows an assembly in which a single visual input signals if there is currently an object at the fovea and hence indicates if gaze is currently fixated on a visual target or on free-space. This input will be used to block, or gate, the

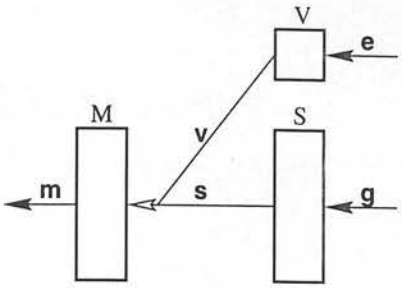


Figure 6.18: The assembly used to learn a gated sensory-motor mapping. The assembly is the same as that used in the previous section (figure 6.9) except an additional input is provided. The action of the motor region is gated by the additional input and the use of nonlinear neurons.

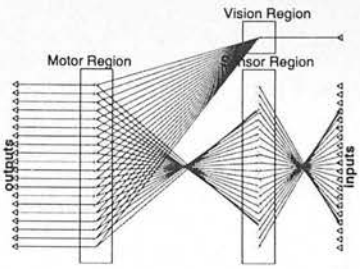


Figure 6.19: Preferred inputs learnt by the nodes. The weight vector of each node was decoded to find the preferred input and this is illustrated by the origin of the line representing the input to each node. Since there is only one input from the vision region all nodes, for which this weight exceeds zero, have the same preferred input, however all these weights had similar strengths. Results are shown after 32000 training cycles, although the weights have a similar appearance from much earlier in training.

performance of the sensory-motor mapping.

The assembly was trained such that the choice of fixation preference changed randomly at every iteration with probability 0.25. Fixation could be directed at either: 1) the hand, in which case sensor input (representing the direction of gaze) corresponded to the motor output; 2) an object, at a random location which remained static until fixation was redirected (hence fixation on an object caused a sequence of the same sensor inputs until the fixation preference changed); 3) fixation on free-space at a random location causing a sequence of random sensor inputs until a new fixation preference was made. In the first two cases the visual input was high, while in the case of looking at free-space the visual input was low. Figure 6.19 shows typical preferred inputs that were learnt in these experiments.

Since nonlinear nodes were used cluster weights were learnt in addition to synaptic weights (in all regions). Most significantly, cluster weights between preferred gaze direction and vision were learnt within the inputs to the motor region. These cluster weights were increased in strength during fixation on the hand or on objects. When gaze was directed towards the hand cluster weights increased between the preferred inputs from the sensor region and the active visual input for the active motor nodes. When gaze was directed towards an object, the mapping between sensor and motor spaces would cause the active motor nodes to correspond to the current sensor inputs and again cause cluster weights to increase between the preferred sensor inputs and the active visual input. However, when gaze was directed towards free-space the active motor nodes would decrease the cluster weights between the inactive visual input and whichever sensor inputs were active, but since the active sensor inputs were chosen randomly these decreases in cluster weights were evenly distributed across all inputs. Strong

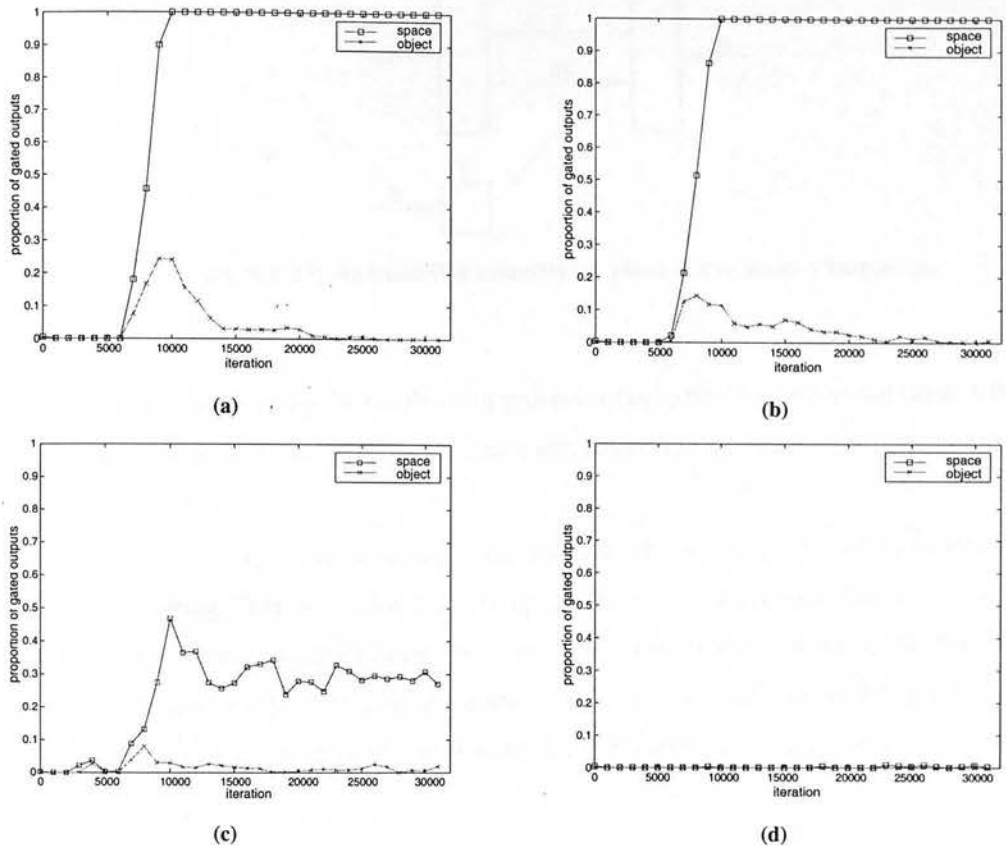


Figure 6.20: The accuracy of the gating of the sensory-motor mapping. The figures show the proportion of occasions that gating occurs, out of all such occasions, over a period of 1000 iterations from the position where the point is plotted. Separate lines show gating on occasions when fixation is on free-space and occasions when fixation is on a target. The percentage of times that each possible type of fixation is made are: (a) objects 50%, free-space 25%, hand 25%; (b) object 33%, free-space 33%, hand 33%; (c) object 25%, free-space 50%, hand 25%; (d) the same conditions as (b) but using linear nodes.

cluster weights were thus formed between the visual input and the preferred sensor inputs to each motor region node whenever gazing at free-space was less frequent than gazing at visual targets. These cluster weights produced gating of the sensor-motor mapping whenever the visual input was low (figure 6.20). This learning is unsupervised, but (as with learning the simple sensory-motor mapping) is reliant on the innate fixation ability. Increasing the proportion of fixations directed towards free-space reduces the correlation between having a high visual input value and the preferred sensor input, and hence can result in failure to learn the appropriate cluster weights to generate gating (figure 6.20(c) shows that unreliable gating is caused when 50% of fixations are towards free-space). The synaptic weights connecting the motor region to the vision region were all similar to each other and similar in strength to the weights from individual nodes in the sensor region. The contribution of the visual input to the activations of the motor nodes is thus small. The reduction in activation caused directly by the lack of visual input was insufficient to cause the input activations of the motor nodes to fall below threshold, and hence this does not account for the gating effect. This can be seen from the results when using linear nodes

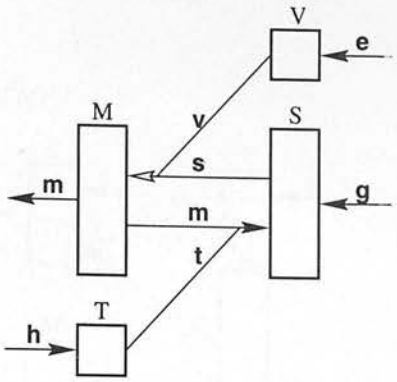


Figure 6.21: An infeasible assembly for gated motor-sensory mappings.

(figure 6.20(d)). In this case the reduction in activation due to the lack of a visual input is the only difference between gazing at a target or gazing at space, rather than any nonlinear interaction in calculating the node activations.

Due to the sensory-motor mapping which has been learnt the agent is likely to touch objects on which it is fixating. This correlation could be used to learn to look at its hand only when it feels touch: touch could be used to gate the mapping from motor to sensor space in the same way that vision is used to gate the mapping from sensor to motor space (figure 6.21). However, as discussed above learning the motor to sensor mapping is infeasible with the current architecture and hence this assembly has not been implemented.

6.2.2.2 Coordinated Behaviour

Useful capacities will require the coordination of many elementary skills. This section considers coordinating the reaching behaviour, learnt above, with the ability to lean forward to reach objects beyond arm's length. For an agent, which has learnt to reach, when attention is focused on an object the mapping from gaze space to arm space will be enabled because there is an object in view, however, no such arm movement will be made unless the gaze direction is within the reachable region. If an object is beyond reach and the agent happened to move forward so that it was brought within reach then the reaching movement would be triggered and it would appear that the agent had purposefully moved forward to touch the object. Such a behaviour would culminate in touch and this signal could be used to provide reinforcement for learning to lean when gaze is directed further forward. Such a reinforcement signal is provided directly from the environment and can be seen as another form of 'suspicious coincidence'. More complex reinforcement signals, abstracted from sensory data using a hierarchy of regions, could also be used when appropriate to the task. The development of behaviour appropriate to particular tasks may require reinforcement in many cases. The role of reinforcement in this situation is to influence the behaviour that is learnt. It has been shown in section 5.6 that apical inputs (section 4.2.4) can be used

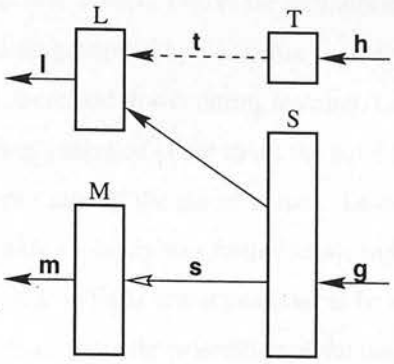


Figure 6.22: The assembly used to learn a coordinated sensory-motor mapping. One sensor region is used to trigger the behaviour of two separate motor regions controlling different actions. A second sensory input is used to provide apical input to one of the motor regions, providing a reinforcement signal to influence the behaviour that is learnt.

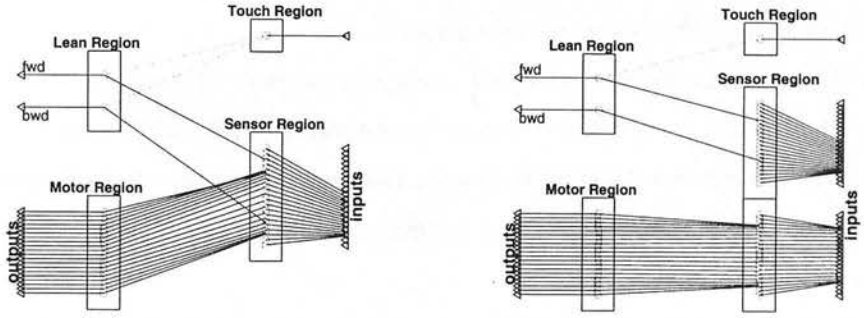


Figure 6.23: Preferred inputs learnt by the nodes. The weight vector of each node was decoded to find the preferred input and this is illustrated by the origin of the line representing the input to each node. Results are shown after 6000 training cycles. (a) For a single sensor region which receives input from the full range of gaze directions. (b) For two separate sensor regions, the upper of which receives input from the full range of gaze directions, while the lower only receives input from one half of the range of gaze directions which are those that correspond to possible motor outputs.

for this purpose. Hence, the assembly shown in figure 6.22 was used to learn this behaviour.

It was found necessary to allow both reaching and leaning to be learnt simultaneously. As before the motor region (controlling the arm) generated hand positions in the range 0 to 1. In contrast to previous experiments, the sensor region input (representing gaze direction) ranged from 0 to 2 so that targets beyond reach (*i.e.*, those at positions greater than 1) could be fixated. The region controlling leaning generated outputs which could shift the absolute hand position by up to one unit. Leaning caused both the arm and the eye to translate an equal distance, so that when fixation was directed towards the hand, leaning had no effect on the sensor input. The mapping between gaze direction and hand position could thus be learnt as before. However, since leaning did affect the position of objects with respect to the eye (and the arm), when fixation was directed towards an object the sensor input did change. When the hand position came within a distance of ± 0.075 from an object a touch input was generated. A new, random, choice for the target for fixation was made with probability 0.25 at each iteration, resulting in

short sequences of gazing at the same target. Targets were chosen so that the hand was fixated 25% of the time, while an object (at a randomly chosen location) was fixated 75% of the time. During training all objects were placed within arm's reach. Due to the slow differentiation of receptive fields the range of hand positions that could be generated by the motor region, and the range of leans that could be generated by the lean region, increased slowly during learning. Leaning increased the range of the hand such that the small leans being generated could cause the hand to touch objects that would otherwise have been beyond the current range of the motor output. Leaning forward would be more likely to cause touching of an object which initially was further away, and conversely leaning backwards would be more likely to cause objects at initially nearer positions to be touched. The touch input, to the apical dendrites of the lean region, thus biases the orientation of the mapping from the gaze region to the lean region. This was achieved since the apical input modifies learning of the previous basal input, hence the results of an action gated learning to only reinforce connections which caused that result. Timing is crucial in this assembly and in order to ensure the correct associations were made the regions were updated in the order: gaze, arm, touch and lean, with the appropriate input to the two sensor regions being calculated just before they were required. The preferred inputs of nodes within this assembly is shown in figure 6.23(a). There are redundant degrees of freedom in this system so that the same location can be reached with a range of different lean and arm positions. However, because actions are performed in series, the reach action can adjust to whatever lean has been made and hence redundancy is not an issue.

During the training phase all objects were randomly placed within reach. Testing was performed by choosing objects to always be positioned such that leaning was required to reach them. Figure 6.24 shows the accuracy of the system for reaching objects during training and testing in comparison with the accuracy when leaning was disabled. The accuracies obtained during training, when objects were always placed within arm's length, were similar with and without leaning. However, during testing, when objects were placed beyond arm's length, there is a considerable difference in the accuracies. Without leaning no objects were successfully touched. With leaning a significant proportion of objects were still touched, however, since reaching an object may require two iterations (one to lean forward and the second to move the arm) the proportion of objects that are touched declines. It can also be seen that the accuracy declines rapidly (figure 6.24(a)). This is due to visual inputs beyond arm's length disrupting the mapping between gaze direction and arm position and making reaching inaccurate. To prevent this from happening an assembly with two separate gaze regions was used, which learnt preferred inputs as shown in figure 6.23(b). The gaze region controlling the arm received input representing gaze directions within the arm's length only, while the second gaze region, controlling leaning, received input encoding the whole range of gaze directions. Using this assembly the accuracy of reaching remained unaffected by the presence of targets beyond arm's reach and the proportion of objects successfully touched remained

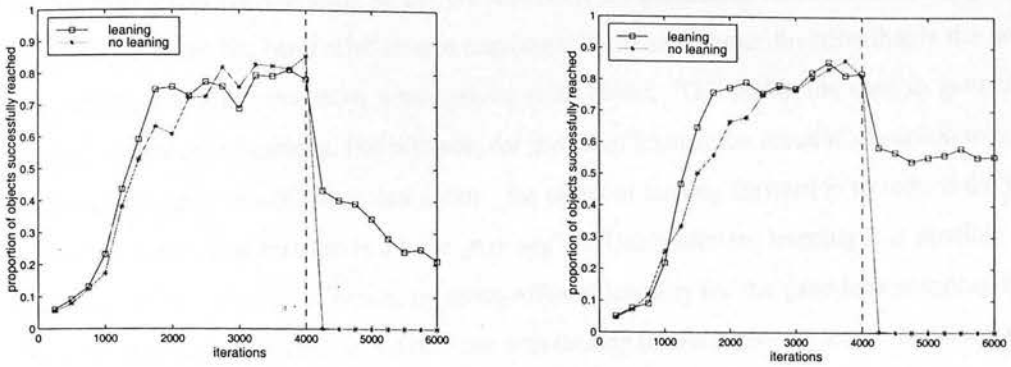


Figure 6.24: The accuracy of reaching and leaning during learning. The figures show the proportion of objects which are successfully reached during intervals of 250 iterations. For the first 4000 iterations all target objects are randomly placed within arm's reach. After 4000 iterations all objects are randomly placed so that leaning is required to reach them. Results are shown for using (a) one single, and (b) two separate gaze regions (corresponding to the assemblies shown in figure 6.23(a) and (b)). Results are averaged over five experiments. The standard deviation over these experiments did not exceed 0.12 for any point in any of the plots.

high during testing (figure 6.24(b)).

This experiment exposes the poor behaviour generated by using open-loop control. An object out of reach will cause the agent to lean forward and touch it. However, moving forward causes the gaze direction required to fixate on the object to reduce which will then trigger leaning backwards at the next iteration. The agent will thus oscillate forwards and backwards when presented with a fixed target. Such behaviour is typical of open-loop control systems.

It was found that learning was only partially stable with only 66%, out of 50 experiments with random variations in the training order, generating the correct mappings. As was found above, the gaze-arm mapping was stable, and most of the problems were due to incorrectly learning the gaze-lean mapping. Touch would appear to provide only a weak bias for learning appropriate RFs in the lean region, and the robustness was found to be particularly sensitive to changes in the experimental details (changing the frequency of fixating on objects, or the distance within which the hand had to come to cause a touch). The results presented here show the best performance obtained, rather than typical results as have been presented for all other experiments.

When learning the gaze-arm mapping (section 6.2.1.1) efferent learning was used. Efferent learning associates actions with their effects, and proved to be very stable. Using afferent learning would be far less stable due to a positive feedback effect: an initial error in the mapping will cause that error to be more likely to be generated in future which will cause the erroneous motor output to be associated with the current sensor input and hence further strengthen the error. Using the same network as in figure 6.9 but with afferent learning between the sensor and motor regions caused all maps to be smooth, and well ordered topologically, but only 54% (about chance level) of experiments with random variations

in the training order, resulted in a correctly orientated mapping. This compares with 98% when using efferent learning (see section 6.2.1.1 page 162). However, the hand-eye coordination task is unusual in that the effects are the same as the preconditions for generating those effects: the gaze direction caused by fixating the hand after an arm movement is the same gaze direction that is the precondition for making that arm movement when gazing at an object. This is not the case in general for other sensory-motor coordinations. For instance, for the act of leaning the result of an action is not the same as the trigger that should cause that action: the effect of leaning forward is to reduce the gaze angle, while the cue to lean forward is a large gaze angle[3]. Using efferent learning it is possible to robustly learn the effects of actions, hence, by using efferent learning for the gaze-lean mapping it should be possible to associate smaller gaze directions with leaning forward. Having learnt the effects of an action it would be possible to compare these potential effects with required goals in order to select an action, *i.e.*, to perform deliberative behaviour. However, where the desired goals, the intentionality, could come from is a major problem. If such deliberative behaviour could be achieved, and occurred sufficiently frequently, then this would provide training data to allow appropriate sensory cues to be associated with triggering those actions. Hence the deliberative behaviour would be automated to form a complex reactive behaviour (see section 3.1.1.3).

6.3 Summary

The model that was implemented had two goals: to learn representations appropriate for controlling behaviour and to learn to associate such representations with appropriate actions. This chapter has demonstrated that the model achieves both of these aims.

[3] This problem would be avoided if a different encoding of the input and output spaces were used. If locations were represented in an absolute coordinate frame, rather than in a relative one, then leaning forward would change the absolute position of the hand. An object at the same absolute position would be an appropriate precondition for leaning forward.

Chapter 7

CONCLUSIONS

7.1 Discussion

This thesis set out to implement a model of development. The aim was to learn from experience appropriate representations for controlling behaviour. What constitutes an appropriate representation will depend on the behaviour that is to be controlled. Direct sensory cues may be available to trigger simple behaviours, while more abstract representations may be required when direct sensory cues are unavailable or to trigger more complex behaviour. As well as being necessary to provide cues for actions the most appropriate level of abstraction is also required to make tractable the task of associating sensory information with those responses. Development is a process in which simple representations and behaviours provide the basis for learning more complex representations and behaviours. In this way more and more abstract representations are formed in order that appropriate levels for learning and controlling different behaviours are found.

A neural network algorithm was implemented as the basis for the model of development. The activation of the nodes in such a network constitutes a representation of the input to the network. The activations of the nodes were modified by a learning algorithm which was used to improve the representation generated in response to future input data. An unsupervised learning algorithm was used. The learning rule caused neurons to respond to correlated sets of inputs. Neurons thus became feature detectors. Because the nodes respond to patterns of input features they encode more abstract features of the input data than are explicitly encoded in the input data itself. In this way simple representations form the basis for learning more complex representations. Furthermore, by modifying the learning rule, to use a trace of the pre-synaptic activity, it was possible for nodes to learn to represent sets of inputs correlated over time, and hence to form invariant representations of features. Such invariant features were learnt from sequences of inputs of random length and without the need to separate the sequences

corresponding to different features.

In its basic form the algorithm is similar to competitive learning. Nodes compete to be selected to represent the current input. The selection is made on the basis of the activation of each node caused by the input. Two methods of calculating this activation have been used: a linear and a nonlinear method. It has been argued that the nonlinear version is more practical and more computationally efficient than previous models of nonlinear nodes. The selection of the winning node is modified by an habituation function which is designed to suppress the competitiveness of nodes which are too frequently active, in order to bias in the long-term all nodes to win the competition an equal number of times. This function thus ensures that a subset of nodes do not come to represent a disproportionately large part of the input space. However, although this means that nodes will be selected with approximately equal frequency it does not necessarily mean that they will be the most active node when selected. Hence, the habituation function does not cause false activations of nodes that are superfluous for representing the data.

The activation of each node will also be affected by the activation of other nodes in the network via lateral inhibition. The winning node is allowed to inhibit all other nodes. A novel form of lateral inhibition is used. This method inhibits the input received by each node (rather than the more common method of inhibiting the output from each node). This form of lateral inhibition enables the network to represent features of the input in a variety of formats. The neural network is able to form distributed and local representations, as well as topologically ordered ones. Hence, it can form representations which are able to encode both single and multiple events in both the discrete and continuous domains, using either topologically or non-topologically organised nodes. The algorithm learns to encode the input data it receives in a format that is appropriate to the structure of that input data. Hence, the most appropriate format for representing the data does not need to be known *a priori*. Nor does the most appropriate number of nodes. In addition, the variety of encodings that can be formed reflects the types of representations found in the cortex. This form of lateral inhibition would thus seem to be computationally attractive and provide a good model of cortical representation, although its implementation is anatomically unjustified.

It has been shown that the algorithm is robust:

- to random variations in the training process,
- to random variations in the parameter values,
- to noise in the input data,
- to noise in the synaptic weight values,
- to the format of encoding of the input,
- to the number of nodes in the network, and,

- to node failure.

It is also computationally tractable and the resulting representations are:

- accurate (and provide generalisation by using multiple active nodes to represent novel input patterns, partially active nodes to represent partial input patterns, and by using interpolation between nodes),
- stable (but are plastic when the input distribution changes, in which case the representation can become realigned to the new data).

The robustness of the algorithm means that parameter tweaking is avoided when the algorithm is applied to different tasks, or is used to model different regions in a larger assembly of neural networks.

The model consists of an assembly of individual neural networks called regions in analogy to cortical regions. Since the algorithm learns encodings which are appropriate formats for the input to other networks, the output from one region can form (part of) the input to other regions. The robustness of the algorithm enables each region to be modelled by a neural network implemented using an identical algorithm. Hence there is uniformity of the algorithm throughout the assembly. Differentiation of regions occurs because of the different input received by each region. The local environment experienced by a region has a strong effect on the representation formed of that environment by that region. There is also uniformity of the algorithm throughout time since the algorithm that learns a representation also generates that representation in response to inputs. A network, that receives its input from a network lower down in the assembly can form a more abstract representation of the sensory data. To achieve this it was found necessary to remove unresponsive nodes from the lower region and to delay learning in the upper region until a stable representation had formed in the lower region. Such processes could correspond to cell death and the wave-of-plasticity observed in the cortex. As well as learning more abstract representations upper regions can be used to learn invariant representations. It was shown that, having learnt an invariant representation for a set of features, an invariant representation of a pattern composed of these features could be learnt without the need to see the pattern in all locations. Hence, the invariant properties of the subfeatures enabled an invariant representation of a pattern to be formed by generalisation rather than learning.

The algorithm used for the individual neural networks in an assembly is an efficient pattern recognition method and has some significant advantages over other neural network algorithms. An assembly of such networks could be used for more complex pattern recognition tasks. Some improvements to the algorithm that may be required to achieve this are suggested in the following section.

It has also been shown how representations can be associated with actions in order to control behaviour. When used to learn sensory-motor associations the algorithm provides exploration of motor effects such that there is no need for a separate training phase. The same algorithm performs explora-

tion (to calibrate the control system) and exploitation (to perform the control task). Hence, the algorithm implemented by each region remains the same throughout time, and can automatically maintain sensor-motor alignment in the face of any changes to the system. In order to learn a more complex skill it was shown that nonlinear nodes could be used to allow one sensor input to gate the effect of another, and that two motor skills could be combined to achieve a single task. In order to develop behaviour appropriate to particular tasks reinforcement may be required. It was shown that such reinforcement signals could be derived from the environment. More abstract reinforcement signals might be derived from more abstract representations formed by the regions in an assembly. Reinforcement is used to modify learning. This was not only used for learning sensory-motor mappings but also for modifying the way in which data was classified in order to find appropriate representations of data.

Two types of learning were used: afferent learning was performed once the new motor outputs were calculated, and thus associated sensory triggers with actions; efferent learning was performed using the previous motor outputs, and thus associated sensory effects with actions. Learning to associate sensory effects with motor actions was found to be extremely robust, even when the sensor inputs frequently failed to correspond to the motor outputs, or when the sensor inputs were inaccurate. Some significant problems were found, however, with learning to associate sensory triggers with appropriate actions. These problems are addressed in the following section. These problems limit the usefulness of the architecture in modelling behavioural development or being used for robot control. However, even as it stands the architecture improves on several similar architectures suggested for robot control (for instance by not requiring a separate training phase).

7.2 Future Work

This section considers improvements that could be made to the model. These improvements can be conveniently separated into 1) relatively minor modifications to the algorithm, and 2) more fundamental extensions to the architecture, which involve using multiple layers of neural networks in each region.

7.2.1 Algorithm Modifications

7.2.1.1 Lateral Inhibition

The competition process between nodes has been approximated by selecting a winning node based on the input activations. Some problems which this approach were identified for ambiguous overlapping patterns (section 5.2) and for generating a coarse code (section 6.2.1.1). It was suggested that a dynamic implementation of the competition process (*i.e.*, equation 4.13) might solve these problems. Such a dynamical implementation may require the use of a more nonlinear transfer function. While it would be

interesting to experiment with such an implementation, the advantages are likely to be more than offset by the increased computational complexity.

Currently, two versions of the algorithm (which differ in the way lateral inhibition is modulated) are required in order to learn both non-topological, and topologically ordered representations. It would be more elegant if the algorithm automatically found appropriate values for σ_I and σ_E to modulate the strength of lateral inhibition. Since topologically ordered data must be represented in the input using coarse coding, it would be possible to select the appropriate values for σ if such coding could be robustly identified. However, modulating the lateral inhibition as a function of distance is not ideal, since it attempts to map the input distribution onto a two-dimensional topology regardless of the dimensionality of the input space. Hence, it might be more appropriate to replace this aspect of the algorithm all together. A further problem with the current implementation is that local excitation and more distant inhibition is generated using the same modulation function. Using separate mechanisms for excitatory and inhibitory connections would allow the net effect of the lateral interaction between two nodes to change sign.

7.2.1.2 Nonlinear Units

The current method of learning cluster weights to implement nonlinear synaptic interactions suffers from three problems. The first problem is that an individual synapse can operate only as part of a single cluster. This problem has been avoided by using multiple nodes acting as a virtual nonlinear unit. But to solve the problem would require allowing inputs to participate in multiple clusters, possibly by forming multiple synapses from the same source. The second problem is that the learning rule has to be modified to prevent negative synaptic weight changes. This is necessary as a node may form several clusters so that a synapse which is not active when the node is active may still form part of another cluster and hence should not have its weight reduced. This results in more poorly differentiated receptive fields, and hence such nodes fail to generate good representations for all the problems that linear nodes have been applied to (*e.g.*, a network of nonlinear units fails to solve the bars problem). The third problem is that no nonlinear unit in a network will respond to a partial input pattern (as might occur if a visual target was partially occluded). Ideally, in this case, the node with the best matching cluster would be partially active, just as the linear node with the best matching RF is partially active for a partial pattern. One solution to all these problems is to use a hierarchy consisting of two regions of linear units instead of a single region of nonlinear units.

7.2.1.3 Hierarchical Abstraction

In the experiments of section 6.1 it was found necessary for lower regions in a hierarchy to have formed stable representations before learning commenced in upper regions. This required that the upper region

was delayed in starting to learn and that any unused nodes were removed from the lower region.

These results suggest that cell death is required for learning more abstract representations. For the current implementation unused nodes were removed by hand. However, it would be desirable to automate the node pruning process, especially for use in larger assemblies. Removing unused nodes was discussed in section 5.1 (see page 112). It is easy to identify unused nodes and hence it would be easy to modify the algorithm to remove them. (It would also be simple to modify the algorithm to insert additional nodes if required by subsequent changes in the input distribution: this is questionable as a biological mechanism, but might provide computational advantages.)

The delay in learning in an upper region could be implemented using a wave-of-plasticity to modulate the learning rate of each region, such that upper regions had very low initial learning rates. However, in complex assemblies determining the order that regions should mature could be difficult. In addition, the delay time would need to be sufficiently long to allow even the slowest representation to form in the lower region, and so the delay time would need to be excessive. Since node activation strength increases from zero during learning another approach would be to wait until the output activations of the lower region exceeded a threshold. This could be achieved by fixing a minimum value for $\tilde{x}$. However, this would also delay learning in other situations in which it is not desirable (such as for the motor region in an assembly learning a sensory-motor mapping). An alternative approach, that would overcome these problems, would be to modify the algorithm so that each neural network could detect when its input representation became stable and only then trigger learning. One indication of stability might be correlations in the activity of sets of inputs *i.e.*, consistent patterns within the input. This criterion would also have the advantage of only enabling learning if there were more abstract representations to be found, and could be used to make embedded patterns more easy to find. Finding correlations in the activity of sets of inputs is exactly what cluster learning (for nonlinear nodes) does. Hence, synaptic learning might be preceded by some form of cluster learning.

It was also found that in some cases (*e.g.*, section 6.1.2.1) it was necessary to use a reduced value of λ_x when finding temporal correlations using a hierarchy of networks. This was due to sensitisation of the inputs to the lower region not only causing input values to persist but also the output activations. Sensitisation of the inputs to the upper region increases the persistence of the outputs of the lower region. At each stage the current input becomes less distinct and there is thus a trade-off between improving discernibility (by increasing λ_x) and the need to impose stronger sensitisation (by decreasing λ_x) in order to learn temporal correlations within the less discernible activation values. Rather than changing the value of λ_x with the number of regions in the hierarchy it would be preferable to allow sensitisation to only affect learning and not activation. This would prevent sensitisation from making the input less distinct at each successive stage.

7.2.1.4 Temporal Considerations

Currently, afferent learning only associates nodes with preconditions that happen at the current time step and efferent learning only associates nodes with postconditions that happen at the following time step. In general, however, it ought to be the case that an event at any time step (in the recent past) can trigger a node activation, and that an effect at any subsequent time step (in the near future) can be associated with a node activation. The trace mechanism (applied to learning invariant representations) does associate recent events with the current node activation. A trace on the node activation could be used to extend the efferent learning rule to associate recent actions with the current input.

However, as well as making such associations it might also be beneficial to store information about the time delay (Barlow, 1996). Hence, the required delay before making an action, or the expected delay before experiencing a result, would be explicitly represented. Neurons would then represent spatio-temporal patterns; they would become spatio-temporal feature detectors. A biologically plausible mechanism for learning such patterns has been reported (Steuber and Willshaw, 1997, 1999). This mechanism uses a learning rule for adjusting a synaptic delay in addition to one adjusting synaptic efficiency. This could be applied to representing frequently used motor sequences with a single node. Such a node would thus represent a higher-level motor skill, and could be used to control this higher-level behaviour. In order to do so, the activation of this node would need to trigger lower-level actions in the correct sequence and with appropriate delays. This would also require the lower-level nodes to learn appropriate delays in their connections with the upper-level node.

Such a mechanism could be used for performing delayed response tasks (*e.g.*, reaching for hidden objects). However, evidence suggests that for such tasks subpopulations of neurons maintain activity during the delay period (Desimone, 1996; Jeannerod, 1997; Ungerleider et al., 1998). In such tasks the delay period is *ad hoc*; in contrast the previous mechanism would only remember particular time delays which re-occurred frequently. Hence, both delay learning and maintained activity need to be used, the former for long-term representation of re-occurring spatio-temporal patterns, and the latter for short-term representation of the current delay (*cf.*, synaptic weights provide long-term representation of re-occurring spatial patterns, while neural activations provide short-term representation of the current input pattern). Maintained activation could provide predictions and represent internal goals. Making the activation of neurons less directly dependent on the current sensory input could make behaviour less reactive. To maintain neural activity during a delay period might require recurrent excitation (Munakata et al., 1997).

One further concern is that learning currently associates correlations in high activation values at every time-step. Since, data has been artificially generated and new data presented each time step this has not been a problem. However, in a real application (such as in a physical robot) input data might not change at every iteration and hence the learning algorithm would be applied repeatedly to the same

input data. It might thus be necessary to restrict learning to occasions when novel inputs are detected (Billard and Hayes, 1999), or modify the learning rules to find coincidences in the changes in activation values (Cohen et al., 1996, 1997; Hallam et al., 1994).

7.2.1.5 Binding

Binding is the process of collecting together the subfeatures of multiple stimuli that are simultaneously represented by the activation pattern of a neural network. For example in a network representing shapes and colours which is receiving input from a red square and a blue triangle, it would be necessary to bind together the representations of 'red' and 'square' and to bind together 'blue' and 'triangle' if it is to be possible to distinguish this pattern of activation from one representing a blue square and a red triangle (Thorpe, 1995). Binding must also be possible to unify features represented across distinct regions. The most popular mechanism which has been proposed for binding is the use of phase synchrony in oscillatory representations which are phase shifted from all other representations (von der Malsburg, 1981). However, there is considerable debate about the mechanisms for, or even the need for, binding (Roskies, 1999; Reynolds and Desimone, 1999; Gray, 1999; Singer, 1999; Shadlen and Movshon, 1999; Riesenhuber and Poggio, 1999a).

The implementation presented here has avoided the binding problem by not considering problems in which binding would be necessary. Note that in a purely local code there is no binding problem for the representation within a single network (Thorpe, 1995) but there will still be a binding problem for the representations of multiple networks.

7.2.1.6 Activation

The value $\tilde{x} = \tau_x \bar{x} + (1 - \tau_x)\tilde{x}$ is used as the pre-synaptic learning threshold for all inputs (in contrast each node has an individual value for its post-synaptic threshold). Since the threshold is the same for all inputs the average strength of each input must be approximately similar. Hence, it will be necessary to ensure that inputs coming from different sources (different regions) are approximately the same average strength. However, the strength of activation coming from a region grows during learning. Hence, there will be problems when a region is receiving inputs from multiple regions (which may be at different stages of learning) and especially when a region receives inputs directly from sensors and from another region. Outputs of regions should thus be scaled to be within a predefined range.

This has not been a problem in general since few experiments have made use of assemblies in which inputs converge on the same dendrite and in most of those that do the two source networks are similar and increasing their output activation at the same rate. The only situation in which this was a problem was for the vision gating reach experiment in which the output of the vision region grows significantly more slowly than the output of the gaze region. In this case the problem was overcome by scaling the

output of the vision region to keep pace with the output of the other region.

7.2.2 Multi-Layer Cortical Models

Section 6.2 highlights several problems with the current implementation.

- It is not possible to form reciprocal connections between regions.
- The architecture provides only open-loop control.
- Associating triggers with actions is unreliable.

All of these problems might be solved by using a model in which cortical regions have multiple layers, rather than the single layer that is currently used.

The cortical sheet is conventionally classified into six distinct neural layers (figure 3.1, section 3.2.1). In simple terms, the upper layers (II, III and IV) can be said to constitute the feedforward pathway, since they receive input from the upper layers of regions lower in the hierarchy and send output to the upper layers of higher regions. Similarly, the lower layers (V and VI) are the feedback pathway since they receive input from the lower layers of regions higher in the hierarchy and send output to the lower layers of lower regions, with additional feedback being provided to layer I (see figure 3.4). While nearly all artificial neural networks investigate how representations of sensory data may be formed in the feedforward pathways, only a few recent models consider the role of feedback (Frey et al., 1997; Hinton et al., 1995; Hinton and Ghahramani, 1997; Rao and Ballard, 1997; Rao, 1999; Lee et al., 1998; O'Reilly, 1998) or propose a theory for the roles of the multiple layers in the cortex (Mumford, 1992; Ebdon, 1996; Grossberg, 1999; Gaudiano and Grossberg, 1991; Barlow, 1994; James and Hoang, 1993; Miller, 1996; Fuentes et al., 1996; Rao and Ballard, 1999; Douglas et al., 1989). Most of these theories are inadequate as they make no attempt to understand how the cortex performs any other task except representing sensory input (Barlow, 1994). In addition, none of these models considers the development of the cortex.

A model consisting of at least two layers of artificial neurons (probably more) would be needed to model the feedforward and feedback pathways. Since sensor inputs arrive in the upper layers and motor outputs are generated in the lower layers the current model uses its single layer inconsistently to model a feedforward layer in sensory regions and a feedback layer in motor regions. The architecture of neural network presented in this thesis should be capable of being used as the basis for modelling each of the layers in a multi-layer model.

7.2.2.1 Multiple Pathways

The current problem of not being able to form reciprocal connections between regions (*e.g.*, figure 6.17) disappears if there are separate feedforward and feedback pathways. A similar need for reciprocal connections occurs in a hierarchical assembly as in figure 4.8(b). In these situations, a single region is receiving two different sources of information, which may possibly be contradictory (*e.g.*, one may represent the current sensory input and the other represent the expected input, or one may represent the actual motor output and the other represent the required output). Obviously, a single neural network is not designed to deal with these different information sources. However, in a multi-layer model each source can provide input to a separate neural network. The question is then how do these information streams interact? How does the activity in each pathway influence the information propagated along the other pathway? And how does each layer affect learning in the other layer?

7.2.2.2 Learning

In a hierarchy, such as that shown in figure 4.8(b), the synapses from the top-down connections impinging on the lower region should not start learning until the upper region forms a stable representation. Hence, there is the same requirement for learning to be delayed in the feedback pathway as has already been seen in the feedforward pathway of an upper region receiving input from a lower region. The delay period before commencing learning will thus vary for the feedforward and feedback layers of the same region. This added complexity provides even more reason for implementing a mechanism, as discussed above, for allowing each network to detect when its input is stable in order to trigger learning. It is known that pyramidal cells in layer VI do not develop until cells in layers II, III and V cooperate in a consistent fashion (Miller, 1996).

Feedback signals might be used to modify learning in the feedforward layer in order to refine the receptive fields of these nodes to provide improved feature detectors for subsequent regions (Rolls and Treves, 1998). A mechanism involving the apical dendrite could be responsible (Rolls and Treves, 1998), or a mechanism in which activations in the feedback layers provide error information for the feedforward layers (O'Reilly, 1998; Hinton and Ghahramani, 1997).

7.2.2.3 Activation

As well as long-term effects on learning it would be expected that the interaction between the layers had short-term effects on the activation. The feedback pathway may affect the current representation of sensory data in the feedforward layers (influence recognition), while the feedforward pathway may affect the motor output from the feedback layers (influence control). Top-down influences will provide information about the expected state of the sensory input to the upper layers as well as the required state

for the motor output in the lower layers: it provides expectation and intention. In humans "the distinction between merely expecting something to happen and intending it to happen may not be made until about 5 months" (Willatts, 1989). This suggest that appropriate connections for the top-down influence on recognition may be learnt before those influencing control.

Effect on Recognition Top-down information should influence the features extracted at lower-levels (Schyns et al., 1998). Such an influence would make perception an active rather than a purely passive process (König and Luksch, 1998). However, the influence of feedback on the feedforward layers is unknown. Several theories suggest that top-down information is subtracted from the feedforward pathway (Mumford, 1992; Barlow, 1994; Rao and Ballard, 1999), while other theories suggest the opposite and require feedback to provide activation of features which match top-down expectations (Grossberg, 1999; Phillips and Singer, 1997; Rolls and Treves, 1998). Both types of model claim neurophysiological support.

In the former case data encoded in the feedback is suppressed in the feedforward pathway so that only unpredicted data is propagated (Mumford, 1992; Barlow, 1994; Rao and Ballard, 1999, 1997; Rao, 1999). Hence, expected events are represented by the absence of firing activity. This is at odds with single cell recordings that show that cells do respond to events (Kock and Poggio, 1999). In addition, the absence of an event will also be represented by no activity which suggests that the representation will be ambiguous.

In the latter case data encoded in the feedback is enhanced in the feedforward pathway (and unpredicted data is suppressed) so that only predicted data is propagated (Grossberg, 1999). However, if unpredicted information is too strongly suppressed then the internal representations are never updated, and significant mismatches between the feedforward and feedback data (which probably indicates that the internally generated expectations are wrong) will be ignored. The unpredicted feedforward data should thus be only weakly inhibited. Cortico-thalamic feedback appears to operate in this way (Rauschecker, 1998). Such enhancement of expected information might provide filling in, pattern completion, the resolution of ambiguous data, contextual guidance, segmentation, binding, recall, priming, and attention (Rolls and Treves, 1998; Phillips and Singer, 1997). Similar functional roles are often proposed for the long-range horizontal connections intrinsic to a region (Grossberg et al., 1996; Usher et al., 1996; Li, 1997, 1998; Phillips and Singer, 1997; Marshall and Srikanth, 1997; Martin and Marshall, 1993; Sirosh and Miikkulainen, 1996a; Frisone et al., 1997; Phillips and Singer, 1997; Somers et al., 1998; Toth et al., 1996; Stemmler et al., 1995; Gilbert et al., 1996). It will be necessary to differentiate these bottom-up effects due to horizontal interactions from the top-down effects due to feedback (Gilbert et al., 1996).

Top-down influence may be required to process ambiguous sensory data, while familiar stimuli can be processed, more rapidly, bottom-up only (Wallis, 1994). Hence, the recognition process can become

automatic in much the same way that behavioural control becomes automated (section 3.1.1.3).

Effect on Control As the cortex learns to control movements it becomes necessary for it to be able to inhibit subcortical control of innate behaviours. Similarly, regions higher up the cortical hierarchy will represent more complex behaviours and must be able to inhibit simpler behaviours, in lower regions, from being activated[1]. It would thus seem necessary for there to be top-down inhibition through the feedback pathway, in addition to the top-down excitation necessary to generate the required actions. The model should be tested using an application in which more complex behaviour is learnt at a higher level, and is expressed using top-down connections, however, this has not been done since it would require reciprocal connections between the motor regions. Top-down influence may cause increased performance errors until suitable top-down connections have been formed. This could explain the increase in performance errors, after initial mastery, that is often observed (Karmiloff-Smith, 1994).

One of the problems encountered in the experiments on learning to lean to reach objects (section 6.2.2.2) was that the model provided only open-loop control (figure 7.1(a)) and hence gave rise to oscillatory behaviour. One possible role for the interaction between cortical layers would be to provide closed-loop control (figure 7.1(b)). In such a system sensor data would provide feedback about the current action which could be compared with the required behaviour (Gaudiano and Grossberg, 1991; Grossberg, 1999). Note that the sensory feedback would derive from perception in the upper cortical layers, called (confusingly) the feedforward layers above. The lower layers would thus provide an (adaptive) inverse control model (figure 7.2(a)). An inverse model estimates the required motor command that will generate a required change in state. The cerebellum is also strongly implicated in the control of movement (Houk et al., 1996). If the lower cortical layers behave as an inverse model then one possible role for the cerebellum would be for it to behave as a forward model (figure 7.2(b)). A forward model predicts the future state based on knowledge of the current state and the motor command. It can be used to overcome time delays in receiving feedback from the environment by providing internally generated feedback to a closed-loop controller.

Another possible role for the cerebellum would be for it to behave as a feedforward (open-loop) controller to complement to role of the cerebrum as a feedback (closed-loop) controller (Parkins, 1997). In this view the cerebellum learns to generate actions that were initially under cortical control (Tyrrell and Willshaw, 1992; Gilbert, 1975; Berthier et al., 1992). Hence, the maturation of the cerebellum relieves the cerebral cortex of the burden of conscious control of movements (Tyrrell and Willshaw, 1992) and this theory proposes that the cerebellum is responsible for automation (section 3.1.1.3, Parkins, 1997)[2].

---

[1] The primary motor cortex is at the bottom of the hierarchy of motor regions (van Essen et al., 1994).

[2] Other theories of cerebellar function suggest that the it provides an inverse model such that it compares the requirements of the output of the motor cortex with the actual movements measured through proprioceptive feedback, and corrects the movement

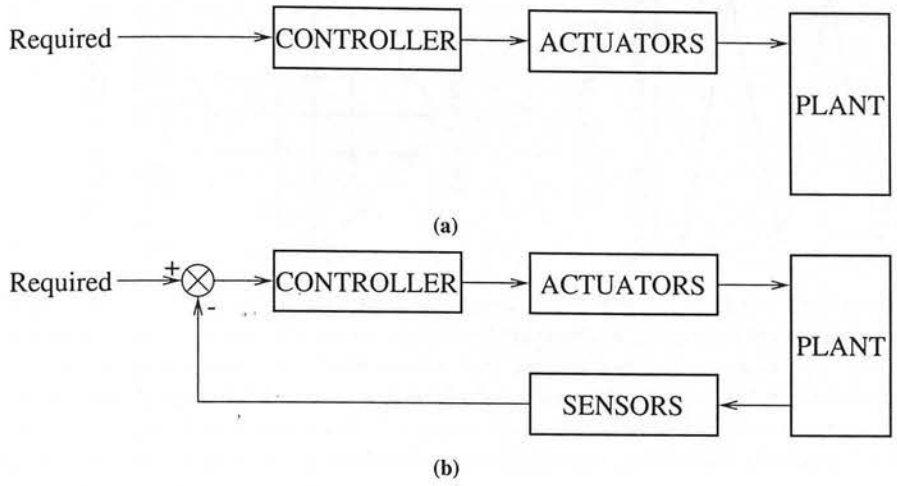


Figure 7.1: Control systems. (a) An open-loop (feedforward) control system. The controller estimates the actions necessary to achieve the required state. If the required state is defined in terms of some other state space than that of the actuators then the controller also performs inverse kinematics. (b) A closed-loop (feedback) control system. Measurements of the current state of the controlled system are fed-back and compared to the required state. The controller attempts to minimise the error between the current and required state. The controller may need to perform inverse kinematics to convert the control signal from the sensor space into the motor space.

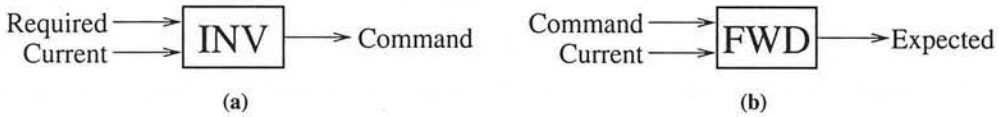


Figure 7.2: Inverse and Forward models for use in control. (a) An inverse model estimates the required motor command that will generate a required change in state. (b) A forward model predicts the future state based on knowledge of the current state and the motor command.

Deliberative behaviour attempts to satisfy internal goals. This will require closed-loop control to compare the required goal with the current state (although the subsequent action, chosen through deliberation, may still be performed under open-loop control)[3]. After extended practise, behaviour that was originally deliberative becomes reactive (through automation) and hence is triggered by external stimuli and is executed in a more open-loop fashion (Parkins, 1997). Due to the lack of multiple layers in the current model closed-loop control could not be achieved and hence deliberative behaviour was not possible. The more complex behaviours that were examined required more abstract triggers to initiate them, but were still purely reactive. It was found that associating triggers directly with actions, using afferent learning, was unreliable (section 6.2.2.2). It would thus seem that trying to directly learn automated complex behaviours, without first using deliberative control, does not work reliably, and the

accordingly. Still other theories of cerebellar function suggest that it provides both inverse and forward models (Kawato, 1995; Kawato et al., 1989; Wolpert and Kawato, 1998; Wolpert et al., 1998).

[3] Note that if the required goal is compared with the expected state generated using a forward model (without producing those actions in the physical world) then this will provide mental simulation and reasoning.

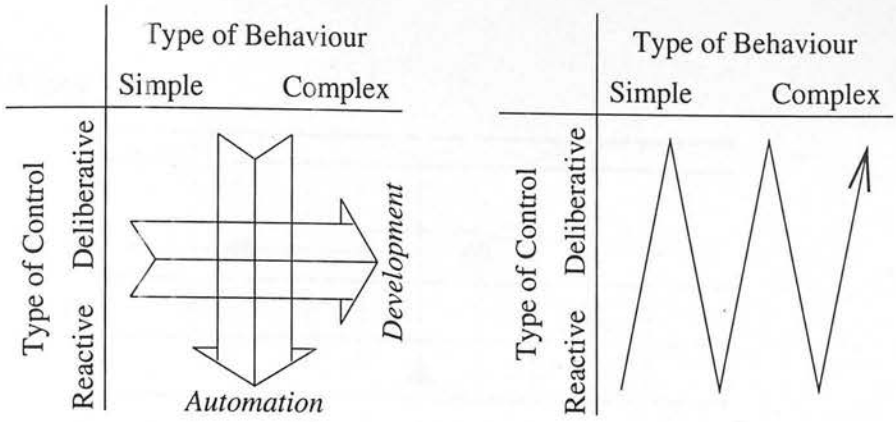


Figure 7.3: Distinction between type of behaviour and form of control. The current model only considers the development of complex behaviour from simple behaviour (simple behaviour due to simple sensory cues and complex behaviour due to more abstract triggers). A multi-layer model would also consider the type of control (reactive behaviour due to open-loop control and deliberative behaviour due to closed-loop control). A reliable developmental progression might require more complex behaviours to initially be under deliberative control, before becoming more reactive and providing the basis for even more complex behaviour to develop.

model ought to be modified to allow deliberative, closed-loop, control (figure 7.3). In contrast to the unreliability of associating triggers directly with actions, it was found that using efferent learning, to associate the effects of actions with node activity, was very robust. Having learnt the effects of neural activations it ought to be possible to compare these effects with the required goals, and hence to perform closed-loop, deliberative, actions. Once the deliberative behaviour is reliable, if it is often performed under the same circumstances, then this will provide training data for associating sensory triggers with the action and hence automate the behaviour. Deliberative control would seem to be required when the representations that are available are at too low a level of abstraction; once representations at the correct level of abstraction have been learnt then the behaviour can be made reactive by generating a direct mapping between these representations.

Achieving closed-loop control will require the cooperation of more regions than when performing open-loop, reactive, behaviour: the cortical regions involved in task performance change such that there are fewer active regions once a task is automated than when it is novel (Petersen et al., 1998; Posner and Rothbart, 1994) (*e.g.*, premotor cortex may be active when learning a skill, but be relatively silent once it has been learnt (Caminiti et al., 1996, p. 174)). Whether or not reactive behaviour arising from automation is controlled by the cerebellum or by recoding in the cerebral cortex (as was suggested in previous chapters) it will be important for the model be able to generate deliberative behaviour in order to learn more complex reactive behaviour.

Information about the required behaviour will come from internal goals (activated by external stimuli), and also directly from sensory input (*e.g.*, the position of the object to be reached for constitutes the required state while the position of the hand constitutes the current state). Closed-loop control ought

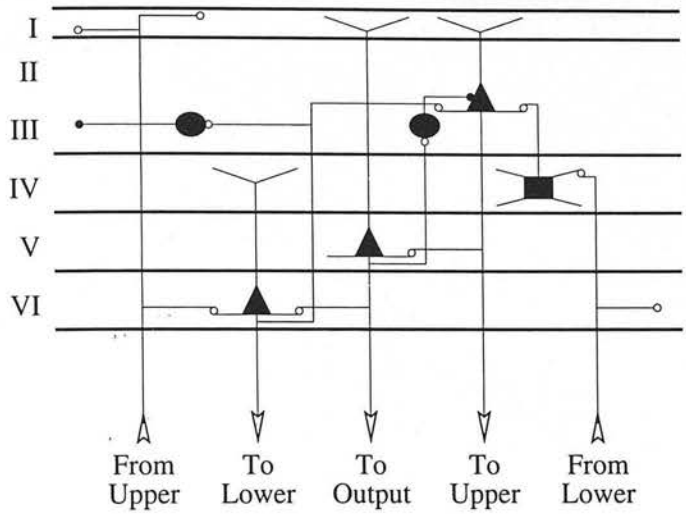


Figure 7.4: A possible multi-layer cortical model. The model should be extended to investigate the interaction between cortical layers. This figure shows a purely speculative cortical circuit for such interaction. It shows dendritic and axonal projections (axons have arrows and circles at terminations) for excitatory cells (pyramidal cell bodies are triangular, spiny-stellate cell bodies are rectangular) and some inhibitory cells (shown by oval cell bodies).

also to be able to bring about arbitrary, *ad hoc*, responses to stimuli, such as in a one-off situation in which an individual is instructed to perform a specific action when an event occurs. In such situations there cannot be a pre-existing direct association between the event and the action.

Figure 7.4 shows a guess at how the cortical circuitry could provide the types of mechanisms that have been discussed. This is purely speculative. It suggests that layer VI pyramidal cells learn the effects of the motor output from layer V cells. Layer VI pyramidals have apical dendrites which ascend only as far as layer IV and could thus learn to associate the sensor input to this layer with previous motor outputs. Layer VI pyramidals also receive top-down inputs which encode the required, or expected, state of the region. The top-down expectation could affect the processing of bottom-up signals either directly via the feedback projections to the apical dendrites in layer I or indirectly via axon collaterals sent from layer VI to the superficial layers Grossberg (1999). The effect of this feedback should be to enhance representations that correspond to the top-down expectation and suppress others. In addition, feedback should modify learning in the feedforward pathway, possibly through the apical dendrites. Bottom-up sensory input will excite cells in layer IV which in turn also excite cells in layer II/III to provide representation of the sensory input. A descending pathway from layer II/III could excite the pyramidals in layer V and hence generate motor outputs. The ascending pathway from layer V to layer II/III is one possible way that the activation of the current motor output could be suppressed, if it is not in agreement with the required output, in order to provide closed-loop control.

The model implemented for this thesis has used single-layer neural networks. There are many potential advantages to using multi-layer neural networks. However, before a useful model of development can be implemented a great deal more research needs to be performed in order to specify what types of multi-layer circuit will be required.

Bibliography

Abbott, L. F. (1994). Decoding neuronal firing and modeling neural networks, *Quarterly Review of Biophysics* **27**: 291–331.

Adorján, P., Levitt, J. B., Lund, J. S. and Obermayer, K. (1998). A model for the intracortical origin of orientation preference tuning in macaque striate cortex, *Technical Report 98-1*, Fachbereich Informatik, Technische Universität Berlin.

Ahalt, S. C. and Fowler, J. E. (1993). Vector quantization using artificial neural network models, *in* D. Docampo and A. R. Figueras (eds), *Proceedings of the International Workshop on Adaptive Methods and Emergent Techniques for Signal Processing and Communications*, pp. 42–61.

Ahalt, S. C., Krishnamurthy, A. K., Chen, P. and Melton, D. E. (1989). Vector quantization using frequency-sensitive competitive learning neural networks, *Proceedings of the IEEE International Conference on Systems Engineering*, IEEE Press, pp. 131–4.

Ahalt, S. C., Krishnamurthy, A. K., Chen, P. and Melton, D. E. (1990). Competitive learning algorithms for vector quantization, *Neural Networks* **3**: 277–90.

Aitken, A. (1994). Architecture for learning to behave, *in* D. Cliff, P. Husbands, J.-A. Meyer and S. W. Wilson (eds), *From Animals To Animats 3: Proceedings of the Third International Conference on the Simulation of Adaptive Behaviour (SAB94)*, MIT Press, pp. 315–24.

Albus, J. S. (1981). *Brains, Behaviour and Robotics*, BYTE Books.

Almássy, N., Edelman, G. M. and Sporns, O. (1998). Behavioural constraints in the development of neuronal properties: a cortical model embedded in a real-world device, *Cerebral Cortex* **8**: 346–61.

Anderson, R. A., Essich, G. K. and Siegal, R. M. (1985). Encoding of spatial location by posterior parietal neurons, *Science* **230**: 456–8.

Araujo, E. G. and Grupen, R. A. (1996). Learning control composition in a complex environment, *in* P. Maes, M. Mataric, J.-A. Meyer, J. Pollack and S. W. Wilson (eds), *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behaviour (SAB96)*, MIT Press, pp. 333–42.

Arbib, M. A. (1995). Schema theory, *in* M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, MIT Press, pp. 830–4.

Arbib, M. A., Conklin, E. J. and Hill, J. C. (1987). *From Schema Theory to Language*, Oxford University Press.

Arbib, M. A., Iberall, T. and Lyons, D. (1987). Schemas that intergrate vision and touch for hand control, *in* M. A. Arbib and A. R. Hanson (eds), *Vision, Brain, and Cooperative Computation*, MIT Press, pp. 489–510.

Arbib, M. A. and Liaw, J.-S. (1995). Sensorimotor transformations in the worlds of frogs and robots, *Artificial Intelligence* **72**: 53–80.

Atkinson, J. and Braddick, O. (1989). Development of basic visual functions, *in* A. Slater and G. Bremner (eds), *Infant Development*, Lawrence Erlbaum Associates, pp. 7–41.

Balkenius, C. (1992). Schemata, learning and conceptual landscapes. http://www.lucs.lu.se/People/-Christian.Balkenius/Abstracts/SLCL.html.

Balkenius, C. (1993). Neural mechanisms for self-organisation of emergent schemata, dynamical schema processing, and semantic constraint satisfaction, *Technical Report LUCS 23*, Department of Cognitive Science, Lund University, Sweden.

Balkenius, C. (1994). Natural intelligence for autonomous agents, *Technical Report LUCS 29*, Depart-

ment of Cognitive Science, Lund University, Sweden.

Balkenius, C. (1995). Multi-modal sensing for robot control, *in* L. F. Nil lasson and M. B. Bodn (eds), *Current trends in connectionism*, Lawrence Erlbaum, pp. 203–16.

Ballard, D. H. (1991). Animate vision, *Artificial Intelligence* **48**: 57–86.

Ballard, D. H., Hayhoe, M. M., Pook, P. K. and Rao, R. P. N. (1997). Deictic codes for the embodiment of cognition, *Behavioural and Brain Sciences* **20**(4): 723–43.

Barlow, H. B. (1960). The coding of sensory messages, *in* W. H. Thorpe and O. L. Zangwill (eds), *Current Problems in Animal Behaviour*, Cambridge University Press, Cambridge, pp. 331–60.

Barlow, H. B. (1972). Single units and sensation: a neuron doctrine for perceptual psychology?, *Perception* **1**: 371–94.

Barlow, H. B. (1989). Unsupervised learning, *Neural Computation* **1**: 295–311.

Barlow, H. B. (1990). Conditions for versatile learning, Helmholtz's unconscious inference, and the task of perception, *Vision Research* **30**: 1561–71.

Barlow, H. B. (1994). What is the computational goal of the neocortex?, *in* C. Koch and J. L. Davis (eds), *Large-Scale Neuronal Theories of the Brain*, MIT Press, chapter 1.

Barlow, H. B. (1995). The neuron doctrine in perception, *in* M. S. Gazzaniga (ed.), *The Cognitive Neurosciences*, MIT Press, chapter 26.

Barlow, H. B. (1996). Intraneuronal information processing, directional selectivity and memory for spatio-temporal sequences, *Network: Computation in Neural Systems* **7**(2): 251–9.

Baron-Cohen, S. (1996). Is there a normal phase of synaesthesia in development?, *Psyche* **2**.

Bartels, A. and Zeki, S. (1998). The theory of multi-stage integration in the visual brain, *Proceedings of the Royal Society of London. Series B* **265**: 2327–32.

Bates, E. and Elman, J. (1993). Connectionism and the study of change, *in* M. H. Johnson (ed.), *Brain Development and Cognition: A Reader*, Blackwell, pp. 623–41.

Bear, M. F. (1985). Activity-dependent modification of functional circuitry as a possible basis for learning, *in* J.-P. Changeux and M. Konishi (eds), *The Neural and Molecular Bases of Learning: Report of the Dahlem Workshop*, Wiley, pp. 281–300.

Beard, R. M. (1969). *An Outline of Piaget's Developmental Psychology for Students and Teachers*, Basic Books.

Bechtel, W. (1996). Yet another revolution? defusing the dynamical system theorists' attack on mental representations, *Technical Report unnumbered*, Philosophy - Neuroscience - Psychology Program, Department of Philosophy, Washington University.

Becker, S. (1993). Learning to categorize objects using temporal coherence, *in* S. J. Hanson, J. D. Cowan and C. L. Giles (eds), *Advances in Neural Information Processing Systems 5*, Morgan Kaufmann, pp. 361–8.

Becker, S. (1995). Unsupervised learning with global objective functions, *in* M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, MIT Press, pp. 997–1000.

Becker, S. (1996). Mutual information maximization: models of cortical self-organization, *Network: Computation in Neural Systems* **7**: 7–31.

Becker, S. (1997). Learning temporally persistent hierarchical representations, *in* M. C. Mozer, M. I. Jordan and T. Petsche (eds), *Advances in Neural Information Processing Systems 9*, MIT Press, pp. 824–30.

Becker, S. (1999). Implicit learning in 3D object recognition: the importance of temporal context, *Neural Computation* **11**(2): 347–74.

Becker, S. and Plumbley, M. (1996). Unsupervised neural network learning procedures for feature extraction and classification, *International Journal of Applied Intelligence* **6**(3): 185–203.

Beer, R. D., Chiel, H. J. and Sterling, L. S. (1991). A biological perspective on autonomous agent design, *in* P. Maes (ed.), *Designing Autonomous Agents; Theory and Practice from Biology to Engineering and Back*, MIT Press, pp. 169–86.

Bell, A. J. and Sejnowski, T. J. (1997a). Edges are the 'independent components' of natural scenes, *in* M. C. Mozer, M. J. Jordan and T. Petsche (eds), *Advances in Neural Information Processing Systems 9*, MIT Press, pp. 831–7.

Bell, A. J. and Sejnowski, T. J. (1997b). The 'independent components' of natural scenes are edge

filters, *Vision Research* **37**: 3327–38.

Bertenthal, B. I. (1996). Origins and early development of perception, action, and representation, *Annual Review of Psychology* **47**: 431–59.

Berthier, N. E., Singh, S. P., Barto, A. G. and Houk, J. C. (1992). A cortico-cerebellar model that learns to generate distributed motor commands to control a kinematic arm, *in* J. E. Moody, S. J. Hanson and R. P. Lippmann (eds), *Advances in Neural Information Processing Systems 4*, Morgan Kaufmann, pp. 611–8.

Bickhard, M. H. (1995). On why constructivism does not yield relativism, *Journal of Experimental and Theoretical Artificial Intelligence* **5**: 275–84.

Billard, A. and Hayes, G. (1999). DRAMA, a connectionist architecture for control and learning in autonomous robots, *Adaptive Behaviour* **7** (in press).

Bonarini, A. (1996). Symbol grounding and a neuro-fuzzy architecture for multisensor fusion, *Proceedings of the World Automation Conference (WAC-ISRAM 96)*, pp. 75–80.

Bousher, D. (1970). *Introduction to the Anatomy and Physiology of the Nervous System*, Blackwell.

Brady, M. and Hu, H. (1994). The mind of a robot, *Philisophical Transactions of the Royal Society of London. Series A* **349**(1689): 16–28.

Braitenberg, V. (1984). *Vehicles. Experiments in Synthetic Psychology*, MIT Press.

Brodmann, K. (1914). Physiologie des gehirns, *in* P. von Braun (ed.), *Neue Deutsche Chirurgie*, Enke, pp. 85–426.

Brookes, A. (1992). The adaptive nature of 3D perception, *in* J.-A. Meyer, H. L. Roitblat and S. W. Wilson (eds), *From Animals to Animats 2: Proceedings of the Second International Conference on the Simulation of Adaptive Behaviour (SAB92)*, MIT Press, pp. 116–21.

Brooks, R. A. (1985). A robust layered control system for a mobile robot, *AI Memo 864*, AI Laboratory, MIT.

Brooks, R. A. (1986). Achieving artificial intelligence through building robots, *AI Memo 899*, AI Laboratory, MIT.

Brooks, R. A. (1991a). Elephants don't play chess, *in* P. Maes (ed.), *Designing Autonomous Agents; Theory and Practice from Biology to Engineering and Back*, MIT Press, pp. 3–15.

Brooks, R. A. (1991b). Intelligence without reason, *AI Memo 1293*, AI Laboratory, MIT.

Brooks, R. A. (1991c). Intelligence without representation, *Artificial Intelligence* **47**: 139–59.

Brooks, R. A. (1994). Coherent behaviour from many adaptive processes, *in* D. Cliff, P. Husbands, J.-A. Meyer and S. W. Wilson (eds), *From Animals To Animats 3: Proceedings of the Third International Conference on the Simulation of Adaptive Behaviour (SAB94)*, MIT Press, pp. 22–9.

Brooks, R. A. (1997). From earwigs to humans, *Robotics and Autonomous Systems* **20**(2-4): 291–304.

Brooks, R. A., Breazeal, C., Irie, R., Kemp, C. C., Marjanovič, M., Scassellati, B. and Williamson, M. (1998). Alternative essences of intelligence, *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI98)*, AAAI Press, pp. 961–8.

Brooks, R. A. and Mataric, M. J. (1993). Real robots, real learning problems, *in* J. H. Connell and S. Mahadevan (eds), *Robot Learning*, Kluwer Academic Publishers, chapter 8, pp. 193–214.

Brooks, R. A. and Stein, L. A. (1993). Building brains for bodies, *AI Memo 1439*, AI Laboratory, MIT.

Brotchie, P. R., Andersen, R. A., Snyder, L. H. and Goodman, S. J. (1995). Head position signals used by parietal neurons to encode locations of visual stimuli, *Nature* **375**: 232–5.

Brown, A. G. (1991). *Nerve Cells and Nervous Systems: An Introduction to Neuroscience*, Springer-Verlag.

Brown, T. H. and Chattarji, S. (1995). Hebbian synaptic plasticity, *in* M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, MIT Press, pp. 454–9.

Callaway, E. M. (1998). Local circuits in primary visual cortex of the macaque monkey, *Annual Review of Neuroscience* **21**: 47–74.

Calvin, W. H. (1996). *The Cerebral Code: Thinking a Thought in the Mosaics of the Mind*, MIT Press.

Caminiti, R., Hoffmann, K.-P., Lacquaniti, F. and Altman, J. (eds) (1996). *Vision and movement mechanisms in the cerebral cortex*, HFSP, Strasbourg.

Chang, C. and Gaudiano, P. (1997). Neural competitive maps for reactive and adaptive navigation, *Proceedings of the 2nd International Conference on Computational Intelligence and Neuroscience*.

Chang, C. and Gaudiano, P. (1998). Application of biological learning theories to mobile robot avoidance and approach behaviours, *Journal of Complex Systems* **1**.

Changeux, J.-P. and Dehaene, S. (1993). Neuronal models of cognitive development, *in* M. H. Johnson (ed.), *Brain Development and Cognition: A Reader*, Blackwell, pp. 363–402. Originally published in: *Cognition* **33**:63-109 (1989).

Charles, D. and Fyfe, C. (1997). Discovering independent sources with an adapted PCA neural network, *in* D. W. Pearson (ed.), *Second International Symposium on Soft Computing, Fuzzy Logic, Artificial Neural Networks, Genetic Algorithms (SOCO97)*, ICSC Press.

Charles, D. and Fyfe, C. (1998). Modelling multiple cause structure using rectification constraints, *Network: Computation in Neural Systems* **9**(2): 167–82.

Chien, S. A., Gervasio, M. T. and DeJong, G. F. (1991). On becoming decreasingly reactive: learning to deliberate minimally, *in* L. A. Birnbaum and G. C. Collins (eds), *Machine Learning: Proceedings of the Eighth International Workshop*, Morgan Kaufmann, pp. 288–92.

Churchland, P. M. (1990). Some reductive strategies in cognitive neurobiology, *in* M. A. Boden (ed.), *The Philosophy of Artificial Intelligence*, Oxford University Press, chapter 14.

Churchland, P. S. (1986). *Neurophilosophy*, MIT Press, chapter 10.

Clark, A. (1994). Autonomous agents and real-time success: some foundational issues, *Technical Report un-numbered*, Philosophy - Neuroscience - Psychology Program, Department of Philosophy Washington University, St. Louis Missouri.

Clark, A. (1997). *Being There: Putting Brain, Body, and World Together Again*, MIT Press.

Clark, A. (1998). Twisted tales: causal complexity and cognitive scientific explanation, *Minds and Machines* **8**(1): 79–99.

Clark, A. (1999a). An embodied cognitive science?, *Trends in Cognitive Sciences* **3**(9): 345–51.

Clark, A. (1999b). Where brain, body, and world collide, *Journal of Cognitive Systems Research* **1**: 38–50.

Clark, A. and Chalmers, D. (1995). The extended mind, *Technical Report unnumbered*, Department of Philosophy, Washington University.

Clark, A. and Thornton, C. (1997). Trading spaces: computation, representation and the limits of uninformed learning, *Behavioural and Brain Sciences* **20**(1): 57–66.

Cliff, D. (1991). Computational neuroethology: a provisional manifesto, *in* J.-A. Meyer and S. W. Wilson (eds), *From Animals to Animats: Proceedings of the First International Conference on the Simulation of Adaptive Behaviour (SAB91)*, MIT Press, pp. 29–39.

Cliff, D., Harvey, I. and Husbands, P. (1992). Incremental evolution of neural network architectures for adaptive behaviour, *Technical Report CSRP 256*, School of Cognitive and Computing Sciences, University of Sussex.

Clocksin, W. F. and Moore, A. W. (1989). Experiments in adaptive state-space robotics, *in* A. G. Cohn (ed.), *Proceedings of the 7th Joint International Conference for the Study of Artificial Intelligence and Simulation of Behaviour (AISB89)*, Pitman, pp. 115–25.

Cohen, M. A. and Grossberg, S. (1986). Neural dynamics of speech and language coding: developmental programs, perceptual grouping, and competition for short term memory, *Human Neurobiology* **5**: 1–22.

Cohen, M. A. and Grossberg, S. (1987). Masking fields: a massively parallel neural architecture for learning, recognizing, and predicting multiple groupings of patterned data, *Applied Optics* **26**: 1866–91.

Cohen, P. R., Atkin, M. S., Oates, T. and Beal, C. R. (1997). Neo: learning conceptual knowledge by sensorimotor interaction with an environment, *Proceedings of the First International Conference on Autonomous Agents*, ACM Press, pp. 170–7.

Cohen, P. R., Oates, T., Atkin, M. S. and Beal, C. R. (1996). Building a baby, *Proceedings of the 18th Annual Conference of the Cognitive Science Society*, pp. 518–22.

Coiton, Y., Gilhodes, J. C., Velay, J. L. and Roll, J. P. (1991). A neural network model for the intersensory coordination involved in goal-directed movements, *Biological Cybernetics* **66**(2): 167–76.

Colombetti, M., Dorigo, M. and Borghi, G. (1996). Behavior analysis and training: a methodology for behavior engineering, *IEEE Transactions on Systems, Man, and Cybernetics – Part B* **26**(3): 365–80.

Connell, J. H. and Mahadevan, S. (1993). Introduction to robot learning, *in* J. H. Connell and S. Mahadevan (eds), *Robot Learning*, Kluwer Academic Publishers, chapter 1, pp. 1–17.

Cowan, J. D. and Friedman, A. E. (1995). Development and regeneration of eye-brain maps, *in* M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, MIT Press, pp. 295–9.

Crick, F. (1989). Neural Edelmanism, *Trends in Neuroscience* **12**: 240–8.

Crick, F. and Asanuma, C. (1986). Certain aspects of the anatomy and physiology of the cerebral cortex, *in* D. E. Rumelhart, J. L. McClelland and The PDP Research Group (eds), *Parallel Distributed Processing: Explorations in the Microstructures of Cognition. Volume 2: Psychological and Biological Models*, MIT Press, pp. 333–71.

Das, A. (1997). Plasticity in adult sensory cortex: a review, *Network: Computation in Neural Systems* **8**(2): R33–R76.

Davidsson, P. (1993). A framework for organisation and representation of concept knowledge in autonomous agents, *Scandinavian Conference of Artificial Intelligence*, IOS Press, pp. 183–92.

Davidsson, P. (1995). On the concept of concept in the context of autonomous agents, *Second World Conference on the Fundamentals of Artificial Intelligence*, pp. 85–96.

de Haan, M., Oliver, A. and Johnson, M. H. (1998). A high-density ERP study of infant and adult face processing, *submitted* .

de Sa, V. R. (1994). Learning classification with unlabeled data, *in* J. D. Cowan, G. Tesauro and J. Alspector (eds), *Advances in Neural Information Processing Systems 6*, Morgan Kaufmann, pp. 112–9.

de Schonen, S. and Mancini, J. (1995). About functional brain specialization: the development of face recognition, *Developmental Cognitive Neuroscience Technical Report 95.1*, Centre for Brain and Cognitive Development, Birkbeck College London.

Dedieu, E. and Mazer, E. (1991). An approach to sensorimotor relevance, *in* F. J. Varela and P. Bourgine (eds), *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, MIT Press, pp. 88–95.

DeFilipe, J. (1997). Microcircuits in the brain, *in* J. Mira, R. Mòreno-Díaz and J. Cabestany (eds), *Biological and Artificial Computation: From Neuroscience to Technology. Proceedings of the International Work-Conference on Artificial Neural Networks (IWANN97)*, Springer, pp. 1–14.

Dellaert, F. and Beer, R. (1994). Towards an evolvable model of development for autonomous agent synthesis, *in* R. Brooks and P. Maes (eds), *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, MIT Press, pp. 246–57.

Der, R. and Smyth, D. (1998). Local online learning of coherent information, *Neural Networks* **11**: 909–25.

DeSieno, D. (1988). Adding a conscience to competitive learning, *International Conference on Neural Networks (ICNN88), volume 1*, IEEE Press, pp. 117–24.

Desimone, R. (1991). Face-selective cells in the temporal cortex of monkey, *Journal of Cognitive Neuroscience* **3**: 1–8.

Desimone, R. (1996). Neural mechanisms for visual memory and their role in attention, *Proceedings of the National Academy of Science USA* **93**: 13494–9.

Digney, B. L. (1998). Learning hierarchical control structures for multiple tasks and changing environments, *in* R. Pfeifer, B. Blumberg, J.-A. Meyer and S. W. Wilson (eds), *From Animals to Animats 5: Proceedings of the Fifth International Conference on Simulation of Adaptive Behaviour (SAB98)*, MIT Press, pp. 321–30.

Dorffner, G. and Prem, E. (1993). Connectionism, symbol grounding, and autonomous agents, *Proceedings of the 15th Annual Meeting of the Cognitive Science Society, Boulder, CO*, pp. 144–8.

Douglas, R. J., Martin, K. A. C. and Whitteridge, D. (1989). A canonical microcircuit for neocortex, *Neural Computation* **1**(4): 480–8.

Drescher, G. L. (1986). Genetic AI: translating Piaget into LISP, *AI Memo 890*, MIT AI Laboratory.

Drescher, G. L. (1987). A mechanism for early Piagetian learning, *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI87)*, Vol. 1, Morgan Kaufmann, pp. 290–4.

Drescher, G. L. (1991). *Made-Up Minds: A Constructivist Approach to Artificial Intelligence*, MIT Press.

Duch, W. (1994). A solution to fundamental problems of cognitive sciences, *Technical Report UMK-KMK-TR 1/94*, Department of Computer Methods, Nicholas Copernicus University, Poland.

Durbin, R. and Rumelhart, D. E. (1989). Product units: a computationally powerful and biologically plausible extension to backpropagation networks, *Neural Computation* **1**: 133–42.

Ebdon, M. (1992). The uniformity of cerebral neocortex and its implications for cognitive science, *Technical Report CSRP-228*, School of Cognitive and Computing Sciences, University of Sussex.

Ebdon, M. (1996). *Towards a General Theory of Cerebral Neocortex*, PhD thesis, School of Cognitive and Computing Sciences, University of Sussex.

Edelman, G. M. (1987). *Neural Darwinism: The Theory of Neuronal Group Selection*, Basic Books.

Edelman, S. (1994). Representation without reconstruction, *Computer Vision, Graphics and Image Processing* **60**: 92–4.

Edelman, S. (1996). Representation is representation of similarities, *Technical Report CS96-08*, Dept. of Applied Maths and Computer Science, The Weizmann Institute of Science, Israel.

Edelman, S. (1997). Computational theories of object recognition, *Trends in Cognitive Sciences* **1**(8): 296–304.

Edelman, S. (1998). Representation is representation of similarities, *Behavioural and Brain Sciences* **21**(4): 449–67.

Edelman, S. and Duvdevani-Bar, S. (1995). Similarity, connectionism, and the problem of representation in vision, *Neural Computation* **9**: 701–20.

Edelman, S., Intrator, N. and Poggio, T. (1997). Complex cells and object recognition. http://www.ai.mit.edu/ edelman/mirror/nips97.ps.Z.

Eglen, S. (1997). *Modelling the Development of the Retinogeniculate Pathway*, PhD thesis, School of Cognitive and Computing Sciences, University of Sussex.

Eglen, S., Bray, A. and Stone, J. (1997). Unsupervised discovery of invariances, *Network: Computation in Neural Systems* **8**(4): 441–52.

Eimas, P. D. (1994). Categorisation in early infancy and the continuity of development, *Cognition* **50**: 83–93.

Elman, J. L. (1993). Learning and development in neural networks: the importance of starting small, *Cognition* **48**: 71–99.

Elman, J. L., Bates, E. A., Johnson, M. H., Karmiloff-Smith, A., Parisi, D. and Plunkett, K. (1996). *Rethinking Innateness: A Connectionist Perspective On Development*, MIT Press.

Encyclopaedia Britannica Online (1997). Animal behaviour: the influence of genetics and experience, Encyclopaedia Britannica, Inc. http://www.eb.co.uk:180/bol/topic?eu=119294&sctn=6.

Erwin, E. and Miller, K. D. (1996). Modeling joint development of ocular dominance and orientation maps in primary visual cortex, *in* J. M. Bower (ed.), *Computational Neuroscience: Trends in Research*, Academic Press, pp. 179–84.

Erwin, E., Obermayer, K. and Schulten, K. (1995). Models of orientation and ocular dominance columns in the visual cortex: a critical comparison, *Neural Computation* **7**(3): 425–68.

Eysenck, M. W. and Keane, M. T. (1990). *Cognitive Psychology: A Student's Handbook*, Lawrence Erlbaum Associates.

Fagg, A. H. and Arbib, M. A. (1998). Modelling parietial-premotor interactions in primate control of grasping, *Neural Networks* **11**: 1277–1303.

Feldman, J. A. and Ballard, D. H. (1982). Connectionist models and their properties, *Cognitive Science* **6**: 205–54.

Felleman, D. J. and Van Essen, D. C. (1991). Distributed hierarchical processing in primate cerebral cortex, *Cerebral Cortex* **1**: 1–47.

Felman, J. A. (1990). Computational constraints on higher neural representations, *in* E. L. Schwartz (ed.), *Computational Neuroscience*, MIT Press.

Ferguson, I. A. (1992). Touringmachines: autonomous agents with attitudes, *Technical Report TR 250*, Computer Laboratory, University of Cambridge.

Ferrell, C. and Kemp, C. (1996). An ontogenetic perspective to scaling sensorimotor intelligence, *AAAI Fall Symposium on Embodied Cognition and Action*, AAAI Press, pp. 45–9.

Field, D. J. (1994). What is the goal of sensory coding?, *Neural Computation* **6**(4): 559–601.

Floreano, D. and Mondada, F. (1996). Evolution of plastic neurocontrollers for situated agents, *in* P. Maes, M. Mataric, J.-A. Meyer, J. Pollack and S. W. Wilson (eds), *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behaviour (SAB96)*, MIT Press, pp. 402–10.

Florence, S. L. and Kaas, J. H. (1995). Somatotopy: plasticity of sensory maps, *in* M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, MIT Press, pp. 888–91.

Földiák, P. (1989). Adaptive network for optimal linear feature extraction, *Proceedings of the IEEE/INNS International Joint Conference on Neural Networks*, Vol. 1, IEEE Press, pp. 401–5.

Földiák, P. (1990). Forming sparse representations by local anti-hebbian learning, *Biological Cybernetics* **64**: 165–70.

Földiák, P. (1991). Learning invariance from transformation sequences, *Neural Computation* **3**: 194–200.

Földiák, P. (1992). Models of sensory coding, *Technical Report CUED/F-INFENG/TR91*, Department of Engineering, University of Cambridge.

Földiák, P. (1993). The 'ideal homunculus': statistical inference from neural population responses, *in* F. Eeckman and J. Bower (eds), *Computation and Neural Systems*, Kluwer Academic Publishers, pp. 55–60.

Földiák, P. and Young, M. P. (1995). Sparse coding in the primate cortex, *in* M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, MIT Press, pp. 895–8.

Frégnac, Y. (1995). Hebbian synaptic plasticity: comparative and developmental aspects, *in* M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, MIT Press, pp. 459–64.

Frey, B. J., Dayan, P. and Hinton, G. E. (1997). A simple algorithm that discovers efficient perceptual codes, *in* M. Jenkin and L. R. Harris (eds), *Computational and Psychophysical Mechanisms of Visual Coding*, Cambridge University Press.

Frisone, F., Morasso, P. G. and Sanguineti, V. (1997). Coordinate-free representation of sensorimotor spaces, *Workshop on Self-Organizing Maps (WSOM97)*, pp. 163–8.

Fritzke, B. (1996). Growing self-organising networks - why?, *in* M. Verleysen (ed.), *European Symposium on Artificial Neural Networks*, D-Facto Publishers, pp. 61–72.

Fuentes, O., Rao, R. P. N. and Wie, M. V. (1995). Hierarchical learning of reactive behaviors in an autonomous mobile robot, *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Vol. 5, pp. 4691–5.

Fuentes, U., Ritz, R., Gerstner, W. and Hemmen, J. L. V. (1996). Vertical signal flow and oscillations in a three-layer model of the cortex, *Journal of Computational Neuroscience* **3**: 125–36.

Fukushima, K. (1980). Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, *Biological Cybernetics* **36**(4).

Fyfe, C. (1996). A scale-invariant feature map, *Network: Computation in Neural Systems* **7**(2): 269–75.

Fyfe, C. (1997). A neural net for PCA and beyond, *Neural Processing Letters* **6**(1/2): 33–41.

Fyfe, C. and Baddeley, R. (1995). Finding compact and sparse-distributed representations of visual images, *Network: Computation in Neural Systems* **6**(3): 333–44.

Gadanho, S. C. (1999). *Reinforcement Learning in Autonomous Robots: An Empirical Investigation of the Role of Emotions*, PhD thesis, University of Edinburgh.

Gaudiano, P. and Grossberg, S. (1991). Vector associative maps: unsupervised real-time error-based learning and control of movement trajectories, *Neural Networks* **4**: 147–83.

Georgopoulos, A. P. (1997). Neural networks and motor control, *The Neuroscientist* **3**: 52–60.

Georgopoulos, A. P., Schwartz, A. B. and Kettner, R. E. (1986). Neuronal population coding of movement direction, *Science* **233**: 1416–9.

Ghahramani, Z., Wolpert, D. M. and Jordan, M. I. (1997). Computational models of sensorimotor integration, *in* P. G. Morasso and V. Sanguineti (eds), *Self-Organization, Computational Maps and Motor Control*, Elsevier Press, pp. 117–47.

Gilbert, C. D., Das, A., Ito, M., Kapadia, M. and Westheimer, G. (1996). Spatial integration and cortical dynamics, *Proceedings of the National Academy of Science USA* **93**: 615–22.

Gilbert, P. F. C. (1975). How the cerebellum could memorize movements, *Nature* **254**(5502): 688–9.

Goodhill, G. J. (1993). Topography and ocular dominance: a model exploring positive correlations,

Biological Cybernetics **69**: 109–18.

Goodhill, G. J. and Barrow, H. G. (1994). The role of weight normalization in competitive learning, *Neural Computation* **6**: 255–69.

Goodhill, G. J. and Willshaw, D. J. (1990). Application of the elastic net algorithm to the formation of ocular dominance stripes, *Network: Computation in Neural Systems* **1**(1): 41–59.

Goodhill, G. J. and Willshaw, D. J. (1994). Elastic net model of ocular dominance: overall stripe pattern and monocular deprivation, *Neural Computation* **6**: 615–21.

Gopnik, A. (1993). How we know our minds: the illusion of first-person knowledge of intentionality, *Behavioural and Brain Sciences* **16**: 1–14.

Gray, C. M. (1999). The temporal correlation hypothesis of visual feature integration: Still alive and well, *Neuron* **24**(1): 31–47.

Greenough, W. T., Black, J. E. and Wallace, C. S. (1993). Experience and brain development, *in* M. H. Johnson (ed.), *Brain Development and Cognition: A Reader*, Blackwell, pp. 290–322.

Gregory, R. L. (ed.) (1987). *The Oxford Companion To The Mind.*, Oxford University Press.

Griffin, D. R. (1990). The question of animal awareness, *in* J. Pickering and M. Skinner (eds), *From Sentience to Symbols. Readings on Consciousness*, University of Toronto Press, pp. 90–101.

Grossberg, S. (1999). How does the cerebral cortex work? Learning, attention, and grouping by the laminar circuits of visual cortex, *Spatial Vision* **12**: 163–87.

Grossberg, S., Mingolla, E. and Ross, W. D. (1996). Visual brain and visual perception: how does the cortex do perceptual grouping?, *Technical Report CAS/CNS-TR-96-018*, Department of Cognitive and Neural Systems and Center for Adaptive Systems, Boston University.

Grzywacz, N. M. and Burgi, P.-Y. (1998). Towards a biophysically plausible bidirectional Hebbian rule, *Neural Computation* **10**: 499–520.

Güler, M. and Sahin, E. (1994). A binary-input supervised neural unit that forms input dependent higher-order synaptic correlations, *Proceedings of World Congress on Neural Networks, III*, pp. 730–5.

Hallam, B. E., Hallam, J. C. T. and Hayes, G. M. (1994). A dynamic net for robot control, *DAI Research Paper 694*, University of Edinburgh Department of Artificial Intelligence.

Hallam, J. C. T. (1991). Playing with toy cars: an experiment in real-time control, *DAI Research Paper 527*, Department of Artificial Intelligence, University of Edinburgh.

Hallam, J. C. T. and Malcolm, C. A. (1994). Behaviour: perception, action and intelligence - the view from situated robotics, *Philosophical Transactions of the Royal Society of London. Series A* **349**(1689): 29–42.

Harnad, S. (1987a). Category induction and representation, *in* S. Harnad (ed.), *Categorical Perception; The Groundwork for Cognition*, Cambridge University Press, chapter 18.

Harnad, S. (1987b). Psychophysical and cognitive aspects of categorical perception: A critical overview, *in* S. Harnad (ed.), *Categorical Perception; The Groundwork for Cognition*, Cambridge University Press, chapter 1.

Harnad, S. (1990). The symbol grounding problem, *in* S. Forrest (ed.), *Emergent Computation. Physica D, 42*, MIT/North-Holland, pp. 335–46.

Harnad, S. (1993a). Grounding symbols in the analog world with neural nets, *Think* **2**(1): 12–78.

Harnad, S. (1993b). Symbol grounding is an empirical problem: neural nets are just a candidate component, *Proceedings of the 15th Annual Meeting of the Cognitive Science Society*, Erlbaum.

Harnad, S. (1994). Does the mind piggy-back on robotic and symbolic capacity?, *in* H. Morowitz and J. Singer (eds), *The Mind, the Brain, and Complex Adaptive Systems*, Santa Fe Institute / Addison Wesley.

Harnad, S., Hanson, S. J. and Lubin, J. (1991). Categorical perception and the evolution of supervised learning in neural nets, *in* D. E. Powers and L. Reeker (eds), *AAAI Symposium on Machine Learning of Natural Language and Ontology*.

Harpur, G. and Prager, R. (1994). A fast method for activating competitive self-organising neural networks, *ISANN'94*.

Harpur, G. and Prager, R. (1996). Development of low entropy coding in a recurrent network, *Network: Computation in Neural Systems* **7**(2): 277–84.

Hebb, D. O. (1949). *The Organization of Behavior: A Neuropsychological Theory*, Wiley.

Heiligenberg, W. (1991). The neural basis of behaviour: a neuroethological view, *Annual Review of Neuroscience* **14**: 247–67.

Hexmoor, H. H., Lammens, J. M. and Shapiro, S. C. (1993). An autonomous agent architecture for integrating "unconscious" and "conscious", reasoned behaviors, *Technical Report 93-37*, University of Buffalo.

Heylighen, F. (1991). Modelling emergence, *World Futures: The Journal of General Evolution* **31**: 89–104.

Hiller, M. J. (1993). *The role of chemical mechanisms in neural computation and learning*, PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

Hinton, G. E., Dayan, P., Frey, B. J. and Neal, R. M. (1995). The wake-sleep algorithm for unsupervised neural networks, *Science* **268**: 1158–61.

Hinton, G. E. and Ghahramani, Z. (1997). Generative models for discovering sparse distributed representations, *Philosophical Transactions of the Royal Society of London. Series B* **352**: 1177–90.

Honavar, V. and Uhr, L. (1993). Generative learning structures and processes for generalized connectionist networks, *Information Sciences* **70**(1-2): 75–108.

Honderich, T. (ed.) (1995). *The Oxford Companion to Philosophy*, Oxford University Press.

Houghton, G. and Tipper, S. P. (1996). Inhibitory mechanisms of neural and cognitive control: applications to selective attention and sequential action, *Brain and Cognition* **30**: 20–43.

Houk, J. C., Buckingham, J. T. and Barto, A. G. (1996). Models of the cerebellum and motor learning, *Behavioural and Brain Sciences* **19**(3): 368–83.

Hubel, D. H. (1988). *Eye, Brain, and Vision.*, Scientific American Library.

Humphrys, M. (1996). Action selection methods using reinforcement learning, *in* P. Maes, M. Mataric, J.-A. Meyer, J. Pollack and S. W. Wilson (eds), *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior (SAB96)*, MIT Press, pp. 135–44.

Huttenlocher, P. R. (1993). Morphometric study of human cerebral cortex development, *in* M. H. Johnson (ed.), *Brain Development and Cognition: A Reader*, Blackwell, pp. 112–24.

Hyvärinen, A. (1999). Survey on independent component analysis, *Neural Computing Surveys* **2**: 94–128.

Intrator, N. (1995). Information theory and visual plasticity, *in* M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, MIT Press, pp. 484–7.

Intrator, N. (1996). Neuronal goals: efficient coding and coincidence detection, *Proceedings of the International Conference on Neural Information Processing: Progress in Neural Information Processing (ICONIP96)*, Vol. 1, pp. 29–34.

Intrator, N. and Cooper, L. N. (1992). Objective function formulation of the BCM theory of visual cortical plasticity: statistical connections, stability conditions, *Neural Networks* **5**: 3–17.

Intrator, N. and Cooper, L. N. (1995). BCM theory of visual cortical plasticity, *in* M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, MIT Press, pp. 153–7.

Intrator, N., Edelman, S. and Bülthoff, H. H. (1995). An integrated approach to the study of features in visual recognition, *Network* **6**: 603–18.

Jacobs, R. A. (1999). Computational studies of the development of functionally specialized neural modules, *Trends in Cognitive Sciences* **3**(1): 31–9.

James, M. R. and Hoang, D. B. (1993). Outline of a theory of isocortex, *in* F. H. Eeckman and J. M. Bower (eds), *Computation and Neural Systems: Proceedings of the Computation and Neural Systems Conference*, Kluwer Academic Press, pp. 455–9.

Janowsky, J. S. (1993). The development and neural basis of memory systems, *in* M. H. Johnson (ed.), *Brain Development and Cognition: A Reader*, Blackwell, pp. 665–78.

Jeannerod, M. (1997). *The Cognitive Neuroscience of Action*, Blackwell.

Johnson, M. H. (1992). Imprinting and the development of face recognition: from chick to man, *Current Directions in Psychological Science* **1**: 52–5.

Johnson, M. H. (1993). Constraints on cortical plasticity, *in* M. H. Johnson (ed.), *Brain Development and Cognition: A Reader*, Blackwell, pp. 703–21.

Johnson, M. H. (1997). *Developmental Cognitive Neuroscience: An Introduction*, Blackwell.

Johnson, M. H. (1998). Cortical plasticity in normal and abnormal cognitive development: evidence and working hypotheses, *Developmental Cognitive Neuroscience Technical Report 98.1*, Centre for Brain and Cognitive Development, Birkbeck College London.

Johnson, M. H. (1999). Ontogenetic constraints on neural and behavioral plasticity: evidence from imprinting and face recognition, *Canadian Journal of Experimental Psychology* **53**: 77–90.

Johnson, M. H., Oliver, A. and Shrager, J. (1998). The paradox of plasticity: a constrained plasticity approach to the emergence of representations in the neocortex, *Cognitive Studies* **5**(2): 5–24.

Johnson, M. H. and Vecera, S. P. (1996). Cortical differentiation and neurocognitive development: the parcellation conjecture, *Behavioural Processes* **36**: 195–212.

Joseph, S. R. H. (1996). PhD interim report: theory of adaptive neural growth. ftp://ftp.cogsci.ed.ac.uk/-pub/srhj/interim_report.ps.gz.

Kaas, J. H. (1991). Plasticity of sensory and motor maps in adult mammals, *Annual Review of Neuroscience* **14**: 137–67.

Kaas, J. H., Florence, S. L. and Jain, N. (1997). Reorganization of sensory systems of primates after injury, *The Neuroscientist* **3**: 123–30.

Kalarickal, G. J. (1998). *Theory of cortical plasticity in vision*, PhD thesis, Department of Computer Science, University of North Carolina.

Kandel, E. R., Schwartz, J. H. and Jessell, T. M. (eds) (1995). *Essentials of Neural Science and Behavior*, Appleton & Lange.

Karmiloff-Smith, A. (1993). Self-organization and cognitive change, *in* M. H. Johnson (ed.), *Brain Development and Cognition: A Reader*, Blackwell, pp. 592–618.

Karmiloff-Smith, A. (1994). Precis of beyond modularity - a developmental perspective on cognitive science, *Behavioural and Brain Sciences* **17**(4): 693–706.

Karni, A. (1996). The acquisition of perceptual and motor skills: a memory system in the adult human cortex, *Cognitive Brain Research* **5**: 39–48.

Karni, A. and Bertini, G. (1997). Learning perceptual skills: behavioral probes into adult cortical plasticity, *Current Opinion in Neurobiology* **7**(4): 530–5.

Karni, A., Meyer, G., Jezzard, P., Adams, M. M., Turner, R. and Ungerleider, L. G. (1995). Functional MRI evidence for adult motor cortex plasticity during motor skill learning, *Nature* **377**(6545): 155.

Karni, A., Meyer, G., Rey-Hipolito, C., Jezzard, P., Adams, M. M., Turner, R. and Ungerleider, L. G. (1998). The acquisition of skilled motor performance: fast and slow experience-driven changes in primary motor cortex, *Proceedings of the National Academy of Science USA* **95**: 861–8.

Kawato, M. (1995). Cerebellum and motor control, *in* M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, MIT Press, pp. 172–7.

Kawato, M., Isobe, M. and Suzuki, R. (1989). Hierarchical learning of voluntary movement by cerebellum and sensory association cortex, *in* M. A. Arbib and S. Amari (eds), *Dynamic Interactions in Neural Networks: Models and Data*, Springer-Verlag, pp. 195–214.

Kay, J. and Phillips, W. A. (1994). Activation functions, computational goals and learning rules for local processors with contextual guidance, *Technical Report CWN-15*, Centre for Cognitive and Computational Neuroscience, University of Stirling.

Kenny, A. (1989). *The Metaphysics of Mind*, Oxford University Press.

Kirsh, D. (1991a). Foundations of AI: the big issues, *Artificial Intelligence* **47**: 3–30.

Kirsh, D. (1991b). Today the earwig, tomorrow man?, *Artificial Intelligence* **47**: 161–84.

Klingspor, V. and Morik, K. (1995). Towards concept formation grounded on perception and action of a mobile robot, *in* U. R. R. Dillmann (ed.), *Intelligent Autonomous Systems 4 (IAS-4)*.

Knudsen, E. I., du Lac, S. and Esterly, S. D. (1990). Computational maps in the brain, *in* J. A. Anderson, A. Pellionisz and E. Rosenfeld (eds), *Neurocomputing 2*, MIT Press, chapter 22.

Kock, C. and Poggio, T. (1999). Predicting the visual world: silence is golden, *Nature Neuroscience* **2**(1): 9–10.

Kodjabachian, J. and Meyer, J.-A. (1994). Development, learning and evolution in animats, *in* Gaussier and Nicoud (eds), *From Perception to Action*, IEEE Computer Society Press.

Kohonen, T. (1997). *Self-Organizing Maps*, Springer.

König, P., Bizzi, E., Burgess, N., Franceschini, N., Kilgard, M. P., Oram, M., Sagerer, G. and Scheier, C.

(1998). Group report: representations in natural and artificial systems, *Zeitschrift fr Naturforschung C* **53**: 738–51.

König, P. and Luksch, H. (1998). Active sensing - closing multiple loops, *Zeitschrift fr Naturforschung* **53c**: 542–9.

Krubitzer, L. (1995). The organisation of neocortex in mammals: Are species differences really so different?, *Trends in Neurosciences* **18**(9): 408–17. Special Issue: Evolution and Development of the Cerebral Cortex.

Krulwich, B. (1991). Learning from deliberated reactivity, *in* L. A. Birnbaum and G. C. Collins (eds), *Machine Learning: Proceedings of the Eighth International Workshop*, Morgan Kaufmann, pp. 318–22.

Kuipers, B., Froom, R., Lee, W.-Y. and Pierce, D. (1993). The semantic hierarchy in robot learning, *in* J. H. Connell and S. Mahadevan (eds), *Robot Learning*, Kluwer Academic Publishers, chapter 6, pp. 141–70.

Kujala, T., Alho, K. and Näätänen, R. (2000). Cross-modal reorganization of human cortical functions, *Trends in Neural Sciences* **23**(3): 115–20.

Kushmerick, N. (1997). Software agents and their bodies, *Minds and Machines* **7**: 227–47.

Lee, T. S., Mumford, D., Romero, R. and Lamme, V. A. F. (1998). The role of primary visual cortex in higher level vision, *Vision Research* **38**: 2429–54.

Li, Z. (1997). Visual segmentation without classification in a model of the primary visual cortex, *AI Memo 1613*, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Li, Z. (1998). Pre-attentive segmentation in the primary visual cortex, *AI Memo 1640*, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Longothetis, N. and Sheinberg, D. L. (1996). Visual object recognition, *Annual Review of Neuroscience* **19**: 577–621.

Lyons, D. M. and Arbib, M. A. (1989). A formal model of computation for sensory-based robotics, *IEEE Transactions on Robotics and Automation* **5**(3): 280–93.

Lyons, D. M. and Hendriks, A. J. (1995). Exploiting patterns of interaction to achieve reactive behaviour, *Artificial Intelligence* **73**: 117–48.

Maes, P. (1991). Learning behaviour networks from experience, *in* F. J. Varela and P. Bourgine (eds), *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, MIT Press, pp. 48–57.

Maes, P. (1992). Behaviour-based artificial intelligence, *in* J.-A. Meyer, H. L. Roitblat and S. W. Wilson (eds), *From Animals to Animats 2: Proceedings of the Second International Conference on the Simulation of Adaptive Behaviour (SAB92)*, MIT Press, pp. 2–10.

Maffei, L. (1985). Complex cells control simple cells, *in* D. Rose and V. G. Dobson (eds), *Models of the Visual Cortex*, Wiley, chapter 34.

Malcolm, C., Hallam, J. and Smithers, T. (1989). An emerging paradigm in robot architecture, *DAI Research Paper 447*, Department of Artificial Intelligence, University of Edinburgh.

Malcolm, C. and Smithers, T. (1991). Symbol grounding via a hybrid architecture in an autonomous assembly system, *in* P. Maes (ed.), *Designing Autonomous Agents; Theory and Practice from Biology to Engineering and Back*, MIT Press, pp. 123–44.

Manderick, B. (1991). Selectionist systems as cognitive systems, *in* F. J. Varela and P. Bourgine (eds), *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, MIT Press.

Marcus, G. F. (1998). Can connectionism save constructivism, *Cognition* **66**: 153–82.

Mareschal, D. and Thomas, M. S. C. (2000). Self-organisation in normal and abnormal cognitive development, *in* A. F. Kalverboer and A. Gramsbergen (eds), *Brain and Behaviour in Human Development. A Source Book*, Kluwer Academic Publishers. (forthcoming).

Marjanović, M., Scassellati, B. and Williamson, M. (1996). Self-taught visually guided pointing for a humanoid robot, *in* P. Maes, M. Mataric, J.-A. Meyer, J. Pollack and S. W. Wilson (eds), *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behaviour (SAB96)*, MIT Press, pp. 35–44.

Markman, A. B. and Dietrich, E. (2000). In defense of representation, *Cognitive Psychology* **40**: 138–71.

Marr, D. (1970). A theory for cerebral neocortex, *Proceedings of the Royal Society of London. Series B* **176**: 161–234.

Marr, D. (1985). Vision: The Philosophy and the Approach, *in* A. M. Aitkenhead and J. M. Slade (eds), *Issues in Cognitive Modelling*, LEA.

Marshall, J. A. (1995a). Adaptive perceptual pattern recognition by self-organizing neural networks: context, uncertainty, multiplicity, and scale, *Neural Networks* **8**(3): 335–62.

Marshall, J. A. (1995b). Motion perception: self-organization, *in* M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, MIT Press, pp. 585–8.

Marshall, J. A. and Gupta, V. S. (1998). Generalization and exclusive allocation of credit in unsupervised category learning, *Network: Computation in Neural Systems* **9**(2): 279–302.

Marshall, J. A. and Kalarickal, G. J. (1996). Modeling dynamic receptive field changes in primary visual cortex using inhibitory learning, *in* J. M. Bower (ed.), *Computational Neuroscience (CNS96)*, Plenum Press.

Marshall, J. A. and Srikanth, V. (1997). Curved trajectory prediction using a self-organised neural network, *submitted* .

Martin, K. E. and Marshall, J. A. (1993). Unsmearing visual motion: development of long-range horizontal intrinsic connections, *in* S. J. Hanson, J. D. Cowan and C. L. Giles (eds), *Advances in Neural Information Processing Systems 5*, Morgan Kaufmann, pp. 417–24.

McClelland, J. L., McNaughton, B. L. and O'Reilly, R. C. (1995). Why the are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory, *Psychological Review* **102**: 419–57.

McClelland, J. L. and Plaut, D. C. (1993). Computational approaches to cognition: top-down approaches, *Current Opinion in Neurobiology* **3**: 209–16.

McClelland, J. L. and Plunkett, K. (1995). Cognitive development, *in* M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, MIT Press, pp. 193–7.

Medler, D. A. (1998). A brief history of connectionism, *Neural Computing Surveys* **1**: 61–101.

Mel, B. W. (1990a). *Connectionist Robot Motion Planning : A Neurally-Inspired Approach to Visually-Guided Reaching*, Vol. 7 of *Perspectives in Artificial Intelligence*, Academic Press.

Mel, B. W. (1990b). The sigma-pi column: a model of associative learning in cerebral cortex, *CNS Memo 6*, Computation and Neural Systems Program, California Institute of Technology.

Mel, B. W. (1992). The clusteron: toward a simple abstraction for a complex neuron, *in* J. E. Moody, S. J. Hanson and R. P. Lippmann (eds), *Advances in Neural Information Processing Systems 4*, Morgan Kaufmann, pp. 35–42.

Mel, B. W. (1994). Information processing in dendritic trees, *Neural Computation* **6**: 1031–85.

Mel, B. W. (1999). Why have dendrites? a computational perspective, *in* G. Stuart, N. Spruston and M. Hausser (eds), *Dendrites*, Oxford University Press.

Mel, B. W. and Koch, C. (1990). Sigma-pi learning: on radial basis functions and cortical associative learning, *in* D. S. Touretzsky (ed.), *Advances in Neural Information Processing Systems 2*, Morgan Kaufmann, pp. 474–81.

Mel, B. W., Ruderman, D. L. and Archie, K. A. (1998). Translation-invariant orientation tuning in visual 'complex' cells could derive from intradendritic computations, *Journal of Neuroscience* **18**: 4325–34.

Metta, G., Sandini, G. and Konczak, J. (1997). Sensori-motor development in biological and artificial systems: on the ontogenesis of goal-directed reaching, *Technical Report LIRA-TR 4/97*, Laboratory for Integrated Advanced Robotics, Dept. of Informatics, Systems and Telecommunications, University of Genova.

Michie, D. (1995). Problem decomposition and the learning of skills, *in* N. Lavrac and S. Wrobel (eds), *Lecture Notes in Artificial Intelligence*, Vol. 912, Springer-Verlag, pp. 17–31.

Miikkulainen, R., Bednar, J. A., Choe, Y. and Sirosh, J. (1997). Self-organization, plasticity, and low-level visual phenomena in a laterally connected map model of the primary visual cortex, *in* R. L. Goldstone, P. G. Schyns and D. L. Medin (eds), *Psychology of Learning and Motivation, volume 36: Perceptual Learning*, Academic Press, pp. 257–308.

Miller, K. D. (1998). Equivalence of a sprouting-and-retraction model and correlation-based plasticity models of neural development, *Neural Computation* **10**: 529–47.

Miller, K. D. and MacKay, D. J. C. (1994). The role of constraints in Hebbian learning, *Neural Computation* **6**: 100–26.

Miller, R. (1996). Neural assemblies and laminar interactions in the cerebral cortex, *Biological Cybernetics* **75**: 253–61.

Mitchell, T. M. (1990). Becoming increasingly reactive, *Proceedings of the 8th National Conference on AI (AAAI90)*, pp. 1051–9.

Montague, P. R., Gally, J. A. and Edelman, G. M. (1991). Spatial signalling in the development and function of neural connections, *Cerebral Cortex* **1**: 199–220.

Morasso, P. G., Sanguineti, V., Frisone, F. and Perico, L. (1998). Coordinate-free sensorimotor processing: computing with population codes, *Neural Networks* **11**: 1417–28.

Morasso, P. and Sanguineti, V. (1994). Self-organising topographic maps for motor planning, *in* D. Cliff, P. Husbands, J.-A. Meyer and S. W. Wilson (eds), *From Animals To Animats 3: Proceedings of the Third International Conference on the Simulation of Adaptive Behaviour (SAB94)*, MIT Press.

Moshman, D. (1996). Does cognition develop beyond childhood?, *The Genetic Epistemologist* **24**(2).

Mountcastle, V. B. (1998). *Perceptual Neuroscience: The Cerebral Cortex*, Harvard University Press.

Mumford, D. (1992). On the computational architecture of the neocortex II: the role of cortico-cortical loops, *Biological Cybernetics* **66**: 241–51.

Mumford, D. (1994). Neuronal architectures for pattern-theoretic problems, *in* C. Koch and J. L. Davis (eds), *Large-Scale Neuronal Theories of the Brain*, MIT Press, pp. 125–152.

Munakata, Y., McClelland, J. L., Johnson, M. H. and Siegler, R. S. (1997). Rethinking infant knowledge: towards and adaptive process account of successes and failures in object permanence tasks, *Psychological Review* **104**(4): 686–713.

Munro, P. W. (1986). State-dependent factors influencing neural plasticity: a partial account of the critical period, *in* D. E. Rumelhart, J. L. McClelland and The PDP Research Group (eds), *Parallel Distributed Processing: Explorations in the Microstructures of Cognition. Volume 2: Psychological and Biological Models*, MIT Press, chapter 24.

Nehmzow, U. (1994). Autonomous acquisition of sensor-motor couplings in robots, *Technical Report UMCS-94-11-1*, Department of Computer Science, University of Manchester.

Neisser, U. (1987a). From direct perception to conceptual structure, *in* U. Neisser (ed.), *Concepts and Conceptual Development: Ecological and Intellectual Factors in Categorization*, Cambridge University Press, chapter 2.

Neisser, U. (1987b). Introduction: the ecological and intellectual bases of categorization, *in* U. Neisser (ed.), *Concepts and Conceptual Development: Ecological and Intellectual Factors in Categorization*, Cambridge University Press, chapter 1.

Newsome, W. T., Britten, K. H. and Movshon, J. A. (1989). Neuronal correlates of a perceptual decision, *Nature* **341**: 52–4.

Nigrin, A. (1993). *Neural Networks for Pattern Recognition*, MIT Press.

Nudo, R. J., Milliken, G. W., Jenkins, W. M. and Merzenich, M. M. (1996). Use-dependent alterations of movement representations in primary motor cortex of adult squirrel monkeys, *The Journal of Neuroscience* **16**(02): 785–807.

O'Brien, G. and Opie, J. (1999). A connectionist theory of phenomenal experience, *Behavioural and Brain Sciences* **22**(1): 127–48.

Oja, E. (1982). A simplified neuron model as a principal component analyser, *Journal of Mathematical Biology* **15**: 267–73.

Oja, E. (1995). PCA, ICA, and nonlinear Hebbian learning, *Proceedings of the International Conference on Artificial Neural Networks (ICANN95)*, IEE Press, pp. 89–94.

O'Leary, D. D. M. (1993). Do cortical areas emerge from a protocortex?, *in* M. H. Johnson (ed.), *Brain Development and Cognition: A Reader*, Blackwell, pp. 323–37.

Olshausen, B. A., Anderson, C. H. and Essen, D. C. V. (1993). A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information, *The Journal of Neuroscience* **13**(11): 4700–19.

Olshausen, B. A. and Field, D. J. (1996a). Emergence of simple-cell receptive properties by learning sparse code for natural images, *Nature* **381**: 607–9.

Olshausen, B. A. and Field, D. J. (1996b). Natural image statistics and efficient coding, *Network: Computation in Neural Systems* **7**(2): 333–9.

Ooyen, A. V. (1994). Activity-dependent neural network development, *Network* **5**: 401–23.

Oram, M. W., Földiák, P., Perrett, D. I. and Sengpiel, F. (1998). The 'ideal homunculus': decoding neural population signals, *Trends in Neurosciences* **21**(6): 259–65.

O'Reilly, R. C. (1998). Six principles for biologically based computational models of cortical cognition, *Trends in Cognitive Sciences* **2**(11): 455–62.

O'Reilly, R. C. and Johnson, M. H. (1994). Object recognition and sensitive periods: a computational analysis of visual imprinting, *Neural Computation* **6**: 357–89.

O'Reilly, R. C. and McClelland, J. L. (1992). The self-organization of spatially invariant representations, *Technical Report PDP.CNS.92.5*, Department of Psychology, Carnegie Mellon University.

Page, M. (2000). Connectionist modelling in psychology: a localist manifesto, *Behavioural and Brain Sciences* **23** (forthcoming).

Pallbo, R. (1994). Mind as evolution and evolution as such, *Technical Report LUCS 34*, Cognitive Science Department, Lund University, Sweden.

Parkins, E. J. (1997). Cerebellum and cerebrum in adaptive control and cognition: a review, *Biological Cybernetics* **77**(2): 79–87.

Patel, M. J. and Schnepf, U. (1992). Concept formation as emergent phenomena, *in* F. J. Varela and P. Bourgine (eds), *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, MIT Press, pp. 11–20.

Pebody, M. (1995). Learning and adaptivity: enhancing reactive behaviour architectures in real-world interaction systems, *Proceedings of the Third European Conference on Artificial Life*, pp. 679–90.

Perkins, S. and Hayes, G. (1996). Robot shaping - principles, methods and architectures, *Workshop on Learning in Robots and Animals (AISB96)*. Also published as Research Report 795, Department of Artifical Intelligence, University of Edinburgh.

Perrett, D. I. (1996). View-dependent coding in the ventral stream and its consequences for recognition, *in* R. Caminiti, K.-P. Hoffmann, F. Lacquaniti and J. Altman (eds), *Vision and Movement Mechanisms in the Cerebral Cortex*, HFSP, Strasbourg, pp. 142–51.

Perrett, D. I., Hietanen, J. K., Oram, M. W. and Benson, P. J. (1992). Organisation and functions of cells responsive to faces in the temporal cortex, *Philosophical Transactions of the Royal Society of London* **335**: 23–30.

Petersen, S. E., Mier, H. V., Fiez, J. A. and Raichle, M. E. (1998). The effects of practice on the functional anatomy of task performance, *Proceedings of the National Academy of Science USA* **95**: 853–60.

Pfeifer, R. and Verschure, P. (1991). Distributed adaptive control: a paradigm for designing autonomous agents, *in* F. J. Varela and P. Bourgine (eds), *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, MIT Press.

Pfeifer, R. and Verschure, P. F. M. J. (1992). Designing efficiently navigating non-goal directed robots, *in* J.-A. Meyer, H. L. Roitblat and S. W. Wilson (eds), *From Animals to Animats 2: Proceedings of the Second International Conference on the Simulation of Adaptive Behaviour (SAB92)*, MIT Press, pp. 31–9.

Phillips, W. A., Kay, J. and Smyth, D. (1995). The discovery of structure by multi-stream networks of local processors with contextual guidance, *Network: Computation in Neural Systems* **6**(2): 225–46.

Phillips, W. A. and Singer, W. (1997). In search of common foundations for cortical computation, *Behavioural and Brain Sciences* **20**(4): 657–722.

Piaget, J. and Inhelder, B. (1966). *The Psychology of the Child*, Routledge and Kegan Paul.

Plunkett, K., Karmiloff-Smith, A., Bates, E., Elman, J. L. and Johnson, M. H. (1997). Connectionism and developmental psychology, *Journal of Child Psychol. Psychiat.* **38**(1): 53–80.

Posner, M. I., DiGirolamo, G. J. and Fernandez-Duque, D. (1997). Brain mechanisms of cognitive skills, *Consciousness and Cognition* **6**: 267–90.

Posner, M. I. and Rothbart, M. K. (1994). Constructing neuronal theories of mind, *in* C. Koch and J. L. Davis (eds), *Large-Scale Neuronal Theories of the Brain*, MIT Press, chapter 9.

Pouget, A. and Sejnowski, T. J. (1995). Spatial representations in the parietal cortex may use basis

functions, *in* G. Tesauro, D. S. Touretzky and T. K. Leen (eds), *Advances in Neural Information Processing Systems 7*, MIT Press, pp. 157–64.

Prescott, T. (1994). Spatial learning and representation in animals, *in* D. Cliff, P. Husbands, J.-A. Meyer and S. W. Wilson (eds), *From Animals To Animats 3: Proceedings of the Third International Conference on the Simulation of Adaptive Behaviour (SAB94)*, MIT Press, pp. 164–73.

Purves, D., White, L. E. and Riddle, D. R. (1996). Is neural development Darwinian?, *Trends in Neurosciences* **19**(11): 460–4.

Quartz, S. R. (1993). Neural networks, nativism, and the plausibility of constructivism, *Cognition* **48**(3): 223–242.

Quartz, S. R. (1999). The constructivist brain, *Trends in Cognitive Sciences* **3**(2): 48–57.

Quartz, S. R. and Sejnowski, T. J. (1997). The neural basis of cognitive development: a constructivist manifesto, *Behavioural and Brain Sciences* **20**(4): 537–96.

Quinlan, P. T. (1998). Structural change and development in real and artificial neural networks, *Neural Networks* **11**(4): 577–99.

Rao, R. P. N. (1999). An optimal estimation approach to visual perception and learning, *Vision Research* **39**(11): 1963–89.

Rao, R. P. N. and Ballard, D. H. (1997). Dynamical model of visual recognition predicts neural response properties in the visual cortex, *Neural Computation* **9**(4): 721–63.

Rao, R. P. N. and Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects, *Nature Neuroscience* **2**(1): 79–87.

Rauschecker, J. P. (1998). Cortical control of the thalamus: top-down processing and plasticity, *Nature Neuroscience* **1**(3): 179–80.

Recanzone, G. H., Schreiner, C. E. and Merzenich, M. M. (1993). Plasticity of frequency representation in the primary auditory cortex following discrimination training in adult owl monkeys, *Journal of Neuroscience* **13**: 87–103.

Reynolds, J. H. and Desimone, R. (1999). The role of neural mechanisms of attention in solving the binding problem, *Neuron* **24**(1): 19–29.

Riegler, A. (1994a). Constructivist artificial life, and beyond, *in* B. McMullin and N. Murphy (eds), *Proceedings of the Workshop on Autopoiesis and Perception*, Technical report bmcm9401, School of Electronic Engineering, Dublin City University.

Riegler, A. (1994b). Constructivist artificial life: the constructivist-anticipatory principle and functional coupling, *in* J. Hopf (ed.), *Proceedings of the 18th German Conference on Artificial Intelligence (KI-94) Workshop on Genetic Algorithms within the Framework of Evolutionary Computation*, Max-Planck Institute for Informatik, Report MPI-I-94-241, pp. 73–83.

Riegler, A. (1996). Personal communication.

Riekki, J. P. and Röening, J. (1995). Architecture for reactive planning of robot actions, *Proceedings of SPIE - the International Society for Optical Engineering* **2352**: 2–11.

Riesenhuber, M. and Poggio, T. (1998a). Just one view: invariances in inferotemporal cell tuning, *in* M. I. Jordan, M. J. Kearns and S. A. Solla (eds), *Advances in Neural Information Processing Systems 10*, pp. 215–21.

Riesenhuber, M. and Poggio, T. (1998b). Modelling invariances in inferotemporal cell tuning, *AI Memo 1629*, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Riesenhuber, M. and Poggio, T. (1999a). Are cortical models really bound by the "binding problem"?, *Neuron* **24**(1): 87–93.

Riesenhuber, M. and Poggio, T. (1999b). Hierarchical models of object recognition in cortex, *Nature Neuroscience* **2**(11): 1019–25.

Rimey, R. and Brown, C. (1992). Task-orientated vision with multiple bayes nets, *in* A. Blake and A. Yuille (eds), *Active Vision.*, MIT Press, pp. 217–36.

Ring, M. (1992). Two methods for hierarchy learning in reinforcement environments, *in* J.-A. Meyer, H. L. Roitblat and S. W. Wilson (eds), *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behaviour (SAB92)*, MIT Press, pp. 148–55.

Ritter, H., Martinetz, T. and Schulten, K. (1992). *Neural Computation and Self-Organizing Maps. An Introduction*, Addison-Wesley.

Roitblat, H. L. (1991). Cognitive action theory as a control architecture, *in* J.-A. Meyer and S. W. Wilson (eds), *From Animals to Animats: Proceedings of the First International Conference on the Simulation of Adaptive Behaviour (SAB91)*, MIT Press, pp. 444–50.

Rokers, B. (1998). *Nurture, nature, structure: a computational approach to learning*, Master's thesis, Faculty of Philosophy, Utrecht University.

Roland, P. E. and Zilles, K. (1998). Structural divisions and functional fields in the human cerebral cortex, *Brain Research Reviews* **26**(2-3): 87–105.

Rolls, R. T. and Treves, A. (1998). *Neural Networks and Brain Function*, Oxford University Press.

Rosenfield, I. (1992). *The Strange, Familiar and Forgotten: An Anatomy of Consciousness*, Picador.

Roskies, A. L. (1999). The binding problem, *Neuron* **24**(1): 7–9.

Rumelhart, D. E., McClelland, J. L. and The PDP Research Group (eds) (1986). *Parallel Distributed Processing: Explorations in the Microstructures of Cognition. Volume 1: Foundations*, MIT Press.

Rumelhart, D. E. and Zipser, D. (1985). Feature discovery by competitive learning, *Cognitive Science* **9**: 75–112.

Rutkowska, J. C. (1994). Emergent functionality in human infants, *in* D. Cliff, P. Husbands, J.-A. Meyer and S. W. Wilson (eds), *From Animals To Animats 3: Proceedings of the Third International Conference on the Simulation of Adaptive Behaviour (SAB94)*, MIT Press, pp. 179–88.

Rutkowska, J. C. (1995). Can development be designed? What we may learn from the Cog project, *Technical Report CSRP 370*, School of Cognitive and Computer Sciences, University of Sussex.

Rutkowska, J. C. (1996). Reassessing Piaget's theory of sensorimotor intelligence: a view from cognitive science, *in* J. G. Bremner (ed.), *Infant Development: Recent Advances*, Lawrence Erlbaum, pp. 137–162. Also published as Cognitive Science Research Paper 369, School of Cognitive and Computing Sciences, University of Sussex.

Rutkowska, J. C. (1997a). Computation, dynamics and sensory-motor development, *Technical Report CSRP 460*, School of Cognitive and Computing Sciences, University of Sussex.

Rutkowska, J. C. (1997b). What's value worth? constraining unsupervised behaviour acquisition, *in* P. Husbands and I. Harvey (eds), *Proceedings of the Fourth European Conference on Artificial Life*, MIT Press.

Sakata, H. (1996). Coding of 3d features of objects used in manipulation by parietial neurons, *in* R. Caminiti, K.-P. Hoffmann, F. Lacquaniti and J. Altman (eds), *Vision and movement mechanisms in the cerebral cortex*, HFSP, Strasbourg, pp. 55–63.

Salinas, E. and Abbott, L. F. (1995). Transfer of coded information from sensory to motor networks, *Journal of Neuroscience* **15**: 6461–74.

Salinas, E. and Abbott, L. F. (1996). A model of multiplicative neural responses in parietal cortex, **93**: 11956–61.

Salinas, E. and Abbott, L. F. (1997a). Attentional gain modulation as a basis for translation invariance, *in* J. Bower (ed.), *Computational Neuroscience, Trends in Research*, Plenum.

Salinas, E. and Abbott, L. F. (1997b). Invariant visual perception from attentional gain fields, *Journal of Neurophysiology* **77**: 3267–72.

Sanes, J. N. and Donoghue, J. P. (1997). Dynamic motor cortical organization, *The Neuroscientist* **3**: 158–65.

Sastry, P. S., Shah, S., Singh, S. and Unnikrishnan, K. P. (1999). Role of feedback in mammalian vision: a new hypothesis and a computational model, *Vision Research* **39**: 131–48.

Sazonov, V. N. and Wnek, J. (1994). A hypothesis-driven constructive induction approach to expanding neural networks, *in* T. Fawcett (ed.), *Workshop on Constructive Induction and Change of Representation (ML-COLT94)*, pp. 55–9.

Scassellati, B. (1998). Building behaviours developmentally: a new formalism, *AAAI Spring Symposium 'Integrating Robotics Research'*.

Scheier, C. and Pfeifer, R. (1995). Classification as sensory-motor coordination. A case study on autonomous agents, *Proceedings of the Third European Conference on Artificial Life ECAL95*, pp. 657–67.

Scheier, C., Pfeifer, R. and Kunyioshi, Y. (1998). Embedded neural networks: exploiting constraints, *Neural Networks* **11**: 1551–69.

Schnepf, U. (1991). Robot ethology: a proposal for research into intelligent autonomous systems, *in* J.-A. Meyer and S. W. Wilson (eds), *From Animals to Animats: Proceedings of the First International Conference on the Simulation of Adaptive Behaviour (SAB91)*, MIT Press, pp. 465–74.

Schraudolph, N. N. and Sejnowski, T. J. (1992). Competitive anti-Hebbian learning of invariants, *in* J. E. Moody, S. J. Hanson and R. P. Lippmann (eds), *Advances in Neural Information Processing Systems 4*, Morgan Kaufmann, pp. 1017–24.

Schulten, K. and Zeller, M. (1996). Topology representing maps and brain function, *Nova Acta Leopoldina* **72**(294): 133–57.

Schyns, P. G., Goldstone, R. L. and Thibaut, J.-P. (1998). The development of features in object concepts, *Behavioural and Brain Sciences* **21**(1): 1–54.

Sejnowski, T. J. (1977). Storing covariance with nonlinearly interacting neurons, *Journal of Mathematical Biology* **4**: 303–21.

Shadlen, M. N. and Movshon, J. A. (1999). Synchrony unbound: A critical evaluation of the temporal binding hypothesis, *Neuron* **24**(1): 67–77.

Shastri, L., Grannes, D. J., Narayanan, S. and Feldman, J. A. (1999). A connectionist encoding of schemas and reactive plans. http://www.icsi.berkeley.edu/ shastri/psfiles/ulm.ps.gz.

Shatz, C. J. (1996). Patterned neural activity and synaptic transmission in the development of the visual system, *in* A. Konnerth, R. Y. Tsien, K. Mikoshiba and J. Altman (eds), *Coincidence Detection in the Nervous System*, HFSP, Strasbourg, pp. 134–43.

Sheperd, G. M. and Brayton, R. K. (1987). Logic operations are properties of computer-simulated interactions between excitable dendritic spines, *Neuroscience* **21**: 151–66.

Shepherd, G. M. (ed.) (1990). *The Synaptic Organization of the Brain*, Oxford University Press.

Shrager, J. and Johnson, M. H. (1996). Dynamic plasticity influences the emergence of function in a simple cortical array, *Neural Networks* **9**(7): 1119–29.

Shultz, T. R. (1991). Simulating stages of human cognitive development with connectionist models, *in* L. Birnbaum and G. Collins (eds), *Machine Learning: Proceedings of the Eighth International Workshop*, Morgan Kaufmann, pp. 105–9.

Singer, W. (1985). Activity dependent self-organisation of synaptic connections as a substrate of learning, *in* J.-P. Changeux and M. Konishi (eds), *The Neural and Molecular Bases of Learning: Report of the Dahlem Workshop*, Wiley, pp. 301–36.

Singer, W. (1995). Development and plasticity of cortical processing architectures, *Science* **270**: 758–64.

Singer, W. (1999). Neuronal synchrony: A versatile code for the definition of relations?, *Neuron* **24**(1): 49–65.

Sirosh, J. and Miikkulainen, R. (1994). Cooperative self-organization of afferent and lateral connections in cortical maps, *Biological Cybernetics* **71**: 66–78.

Sirosh, J. and Miikkulainen, R. (1995). Ocular dominance and patterned lateral connections in a self-organizing model of the primary visual cortex, *in* G. Tesauro, D. S. Touretzky and T. K. Leen (eds), *Advances in Neural Information Processing Systems 7*, MIT Press, pp. 109–16.

Sirosh, J. and Miikkulainen, R. (1996a). Introduction: the emerging understanding of lateral interactions in the cortex, *in* J. Sirosh, R. Miikkulainen and Y. Choe (eds), *Lateral Interactions in the Cortex: Structure and Function*, http://www.cs.utexas.edu/users/nn/web-pubs/htmlbook96/.

Sirosh, J. and Miikkulainen, R. (1996b). Topographic receptive fields and patterned lateral interaction in a self-organizing model of the primary visual cortex, *Neural Computation* **9**: 577–94.

Sirosh, J., Miikkulainen, R. and Bednar, J. A. (1996). Self-organization of orientation maps, lateral connections, and dynamic receptive fields in the primary visual cortex, *in* J. Sirosh, R. Miikkulainen and Y. Choe (eds), *Lateral Interactions in the Cortex: Structure and Function*, http://www.cs.utexas.edu/-users/nn/web-pubs/htmlbook96/.

Slater, A. (1989). Visual memory and perception in early infancy, *in* A. Slater and G. Bremner (eds), *Infant Development*, Lawrence Erlbaum Associates, pp. 43–71.

Slater, A. and Bremner, G. (eds) (1989). *Infant Development*, Lawrence Erlbaum Associates.

Sloman, A. (1989). On designing a visual system (towards a Gibsonian computational model of vision), *Journal of Experimental and Theoretical Artificial Intelligence* **1**(4): 289–337.

Smithers, T. (1997). Autonomy in robots and other agents, *Brain and Cognition* **34**: 88–106.

Solomon, K. O., Medin, D. L. and Lynch, E. (1999). Concepts do more than categorize, *Trends in Cognitive Sciences* **3**(3): 99–105.

Somers, D. C., Todorov, E. V., Siapas, A. G., Toth, L. J., Kim, D. S. and Sur, M. (1998). A local circuit approach to understanding integration of long-range inputs in primary visual cortex, *Cerebral Cortex* **8**(3).

Somogyi, P. and Martin, K. A. C. (1985). Cortical circuitry underlying inhibitory processes in cat area 17, *in* D. Rose and V. G. Dobson (eds), *Models of the Visual Cortex*, Wiley, chapter 54.

Spratling, M. W. and Hayes, G. M. (1998). A self-organising neural network for modelling cortical development, *in* M. Verleysen (ed.), *6th European Symposium on Artificial Neural Networks (ESANN98)*, D-facto Publications, pp. 333–8.

Steels, L. (1993). Building agents out of autonomous behaviour systems, *in* L. Steels and R. A. Brooks (eds), *The 'Artificial Life' Route to 'Artificial Intelligence'. Building Situated Embodied Agents*, Lawrence Erlbaum.

Stemmler, M., Usher, M. and Niebur, E. (1995). Lateral interactions in primary visual cortex: a model bridging physiology and psychophysics, *Science* **269**: 1877–80.

Steuber, V. and Willshaw, D. J. (1997). How a single Purkinje cell could learn the adaptive timing of the classically conditioned eye-blink response, *in* W. Gerstner (ed.), *Proceedings of the 7th International Conference for Artifical Neural Networks (ICANN97)*, Springer-Verlag, pp. 115–20.

Steuber, V. and Willshaw, D. J. (1999). Adaptive leaky integrator models of cerebellar purkinje cells can learn the clustering of temporal patterns, *Neurocomputing* **26-27**(1-3): 271–6.

Stone, J. and Bray, A. (1995). A learning rule for extracting spatio-temporal invariances, *Network: Computation in Neural Systems* **6**(3): 1–8.

Sun, R., Peterson, T. and Merrill, E. (1996). Bottom-up skill learning in reactive sequential decision tasks, *Proceedings of the Cognitive Science Conference*, Lawrence Erlbaum Associates, pp. 684–90.

Sun, R., Peterson, T. and Merrill, E. (1999). A hybrid architecture for situated learning of reactive sequential decision making, *Applied Intelligence* **11**(1): 109–27.

Sur, M. (1989). Visual plasticity in the auditory pathway: visual inputs induced into auditory thalamus and cortex illustrate principles of adaptive organisation in sensory systems, *in* M. A. Arbib and S. Amari (eds), *Dynamic Interactions in Neural Networks: Models and Data*, Springer-Verlag, pp. 35–51.

Sutton, R. S. (1991). Reinforcement learning architectures for animats, *in* J.-A. Meyer and S. W. Wilson (eds), *From Animals to Animats: Proceedings of the First International Conference on the Simulation of Adaptive Behaviour (SAB91)*, MIT Press, pp. 288–96.

Swindale, N. V. (1996). The development of topography in the visual cortex: a review of models, *Network: Computation in Neural Systems* **7**(2): 161–247.

Tanaka, K. (1996a). A continuous map of higher-level visual features of objects in monkey inferotemporal cortex, *in* A. Konnerth, R. Tsien, K. Mikoshiba and J. Altman (eds), *Coincidence Detection in the Nervous System*, HFSP, Strasbourg, pp. 143–51.

Tanaka, K. (1996b). Inferotemporal cortex and object recognition, *in* R. Caminiti, K.-P. Hoffmann, F. Lacquaniti and J. Altman (eds), *Vision and Movement Mechanisms in the Cerebral Cortex*, HFSP, Strasbourg, pp. 126–33.

Tani, J. (1996). Does dynamics solve the symbol grounding problem of robots?, *Workshop on Learning in Robots and Animals (AISB96)*, pp. 66–74.

Templeman, J. N. and Loew, M. H. (1989). Staged assimilation: a system for detecting invariant features in temporally coherent visual stimuli, *Proceedings of the International Joint Conference on Neural Networks (IJCNN89), volume 1*, IEEE Press, pp. 731–8.

Thompson, I. (1997). Cortical development: a role for spontaneous activity, *Current Biology* **7**: R324–R326.

Thornton, C. (1994a). Emergent representation/green cognition, *Proceedings of DRABC94: On the Role of Dynamics and Representation in Adaptive Behaviour and Cognition*, pp. 192–3.

Thornton, C. (1994b). A general model of the implicit/explicit transition in representational redescription, *Proceedings of the First Australian Conference on Cognitive Science*, p. 16.

Thornton, C. (1994c). Learning where to go without knowing where that is: the acquisition of a non-

reactive mobot behaviour by explicitation, *Technical Report CSPR 361*, School of Cognitive and Computing Sciences, University of Sussex.

Thornton, C. (1995). Brave mobots use representation, *Technical Report CSPR 401*, School of Cognitive and Computing Sciences, University of Sussex.

Thornton, C. (1996a). Re-presenting representation, *in* D. M. Peterson (ed.), *Forms of Representation: An Interdisciplinary Theme for Cognitive Science*, Intellect, pp. 152–62.

Thornton, C. (1996b). Unsupervised constructive learning, *Technical Report CSRP 449*, School of Cognitive and Computing Sciences, University of Sussex.

Thornton, C. (1997). Truth-from-trash learning and the mobot footballer, *Technical Report CSRP 504*, School of Cognitive and Computing Sciences, University of Sussex.

Thorpe, S. J. (1995). Localized versus distributed representations, *in* M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, MIT Press, pp. 549–52.

Toates, F. (1994). What is cognitive and what is not cognitive?, *in* D. Cliff, P. Husbands, J.-A. Meyer and S. W. Wilson (eds), *From Animals To Animats 3: Proceedings of the Third International Conference on the Simulation of Adaptive Behaviour (SAB94)*, MIT Press.

Toates, F. (1998). The interaction of cognitive and stimulus-response processes in the control of behaviour, *Neuroscience and Biobehavioural Reviews* **22**(1): 59–83.

Toth, L. J., Rao, S. C., Kim, D.-S., Somers, D. and Sur, M. (1996). Subthreshold facilitation and suppression in primary visual cortex revealed by intrinsic signal imaging, *Proceedings of the National Academy of Science USA* **93**: 9869–74.

Tovee, M. J., Rolls, E. T. and Azzopardi, P. (1994). Translation invariance in the responses to faces of single neurons in the temporal visual cortical areas of the alert macaque, *Journal of Neurophysiology* **72**(3): 1049–60.

Tsotsos, J. K. (1995). Behaviourist intelligence and the scaling problem, *Artificial Intelligence* **75**: 135–60.

Turing, A. M. (1950). Computing machinery and intelligence, *Mind* **16**(2236): 433–60. Reprinted in Boden, M. A. (ed.) The Philosophy of Artificial Intelligence, Oxford University Press, pp. 40-66 (1990).

Turkewitz, G. and Kenny, P. A. (1993). Limitations on input as a basis for neural organization and perceptual development: a preliminary theoretical statement, *in* M. H. Johnson (ed.), *Brain Development and Cognition: A Reader*, Blackwell, pp. 510–22. Originally published in: *Developmental Psychobiology* **15**:357-68 (1982).

Tyrrell, T. and Willshaw, D. (1992). Cerebellar cortex: its simulation and the relevance of Marr's theory, *Philosophical Transactions of the Royal Society of London* **336**: 239–57.

Umerez, J., Etxeberria, A. and Moreno, A. (1993). Emergence and functionality. ftp://ftp.vub.ac.be/-pub/projects/Principia_Cybernetica/Papers_Others/Umerez|Emergence&Functionality.txt.

Ungerleider, L. G. (1996). What and where in the human brain? evidence from functional brain imaging studies, *in* R. Caminiti, K.-P. Hoffmann, F. Lacquaniti and J. Altman (eds), *Vision and movement mechanisms in the cerebral cortex*, HFSP, Strasbourg, pp. 23–30.

Ungerleider, L. G., Courtney, S. M. and Haxby, J. V. (1998). A neural system for human visual working memory, *Proceedings of the National Academy of Science USA* **95**: 883–90.

Usher, M., Stemmler, M. and Niebur, E. (1996). The role of lateral connections in visual cortex: dynamics and information processing, *in* J. Sirosh, R. Miikkulainen and Y. Choe (eds), *Lateral Interactions in the Cortex: Structure and Function*, http://www.cs.utexas.edu/users/nn/web-pubs/htmlbook96/.

Vaario, J. (1994). Artificial life as constructivist AI, *Journal of SICE (Society of Instrument and Control Engineers)* **33**(1): 65–71.

van Essen, D. C., Anderson, C. W. and Olshausen, B. A. (1994). Dynamic routing strategies in sensory, motor, and cognitive processing, *in* C. Koch and J. L. Davis (eds), *Large-scale neuronal theories of the brain*, MIT Press.

van Essen, D. C. and Deyoe, E. A. (1995). Concurrent processing in the primate visual cortex, *in* M. S. Gazzaniga (ed.), *The Cognitive Neurosciences*, MIT Press, chapter 24.

van Hateren, J. H. and Ruderman, D. L. (1998). Independent component analysis of natural image sequences yields spatio-temporal filters similar to simple cells in primary visual cortex, *Proceedings*

of the Royal Society of London. Series B **265**: 2315–20.

Vera, A. H. and Simon, H. A. (1993). Situated action: a symbolic interpretation, *Cognitive Science* **17**: 7–48.

Verschure, P. F. M. J., Krose, B. J. A. and Pfeifer, R. (1992). Distributed adaptive control: the self-organisation of structured behaviour, *Robotics and Autonomous Systems* **9**: 181–96.

von der Malsburg, C. (1973). Self-organisation of orientation sensitive cells in the striate cortex, *Kybernetik* **14**: 85–100.

von der Malsburg, C. (1981). The correlation theory of brain function, *Technical Report 81-2*, Max-Planck-Institute for Biophysical Chemistry. Reprinted in Domany, E., van Hemmen, J. L. and Schulten K. (eds.) Models of Neural Networks II, Springer-Verlag, pp. 95-119 (1994).

von der Malsburg, C. (1995). Self-organisation and the brain, *in* M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, MIT Press, pp. 840–3.

Wallis, G. (1994). *Neural Mechanisms Underlying Processing in the Visual Areas of the Occipital and Temporal Lobes*, PhD thesis, Corpus Christi College/Department of Experimental Psychology, University of Oxford.

Wallis, G. (1996). Using spatio-temporal correlations to learn invariant object recognition, *Neural Networks* **9**(9): 1513–9.

Wallis, G. (1998). Spatio-temporal influences at the neural level of object recognition, *Network: Computation in Neural Systems* **9**(2): 265–78.

Wallis, G. and Baddeley, R. (1997). Optimal, unsupervised learning in invariant object recognition, *Neural Computation* **9**(4): 959–70.

Wallis, G. and Bülthoff, H. (1999). Learning to recognize objects, *Trends in Cognitive Sciences* **3**(1): 22–31.

Wallis, G., Rolls, E. and Földiák, P. (1993). Learning invariant responses to the natural transformations of objects, *International Joint Conference on Neural Networks*, Vol. 2, pp. 1087–90.

Wallis, G. and Rolls, E. T. (1997). A model of invariant object recognition in the visual system, *Progress in Neurobiology* **51**: 167–94.

Warrick, S. and Fox, M. (1994). Symbol acquisition in an adaptive agent architecture, *Workshop on Models or Behaviours: Which Way Forward For Robotics? (AISB94)*. Also published as Research Note RN/94/13, Department of Computer Science, University College London.

Werner, G. M. (1994). Using second order neural connections for motivation of behaviour choices, *in* D. Cliff, P. Husbands, J.-A. Meyer and S. W. Wilson (eds), *From Animals To Animats 3: Proceedings of the Third International Conference on the Simulation of Adaptive Behaviour (SAB94)*, MIT Press, pp. 154–61.

Willatts, P. (1989). Development of problem-solving in infancy, *in* A. Slater and G. Bremner (eds), *Infant Development*, Lawrence Erlbaum Associates, pp. 143–82.

Williams, H. G. (1983). *Perceptual and Motor Development*, Prentice-Hall.

Wilson, S. W. (1991). The animat path to AI, *in* J.-A. Meyer and S. W. Wilson (eds), *From Animals to Animats: Proceedings of the First International Conference on the Simulation of Adaptive Behaviour (SAB91)*, MIT Press, pp. 15–21.

Wolpert, D. M. and Kawato, M. (1998). Multiple paired forward and inverse models for motor control, *Neural Networks* **11**: 1317–29.

Wolpert, D. M., Miall, R. C. and Kawato, M. (1998). Internal models in the cerebellum, *Trends in Cognitive Sciences* **2**: 338–47.

Yamauchi, B. and Beer, R. D. (1994). Integrating reactive, sequential and learning behavior using dynamical neural networks, *in* J.-A. Meyer, H. L. Roitblat and S. W. Wilson (eds), *From Animals to Animats 2: Proceedings of the Second International Conference on the Simulation of Adaptive Behaviour (SAB92)*, MIT Press, pp. 382–91.

Yuille, A. L. and Geiger, D. (1995). Winner-take-all mechanisms, *in* M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, MIT Press, pp. 1056–60.

Yuille, A. L. and Greynacz, N. M. (1989). A winner-take-all mechanism based on presynaptic inhibition feedback, *Neural Computation* **1**: 334–47.

Ziemke, T. (1997). Rethinking grounding, *in* A. Riegler and M. Peschl (eds), *Does Representation Need*

Reality? Proceedings of the International Conference on New Trends in Cognitive Science (NTCS97), Austrian Society for Cognitive Science, pp. 87–94.

Zipser, D. and Andersen, R. A. (1988). A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons, *Nature* **331**: 679–84.